Additional LyX Features

Version 2.0.x

by the LyX Team*

January 12, 2013

*Principal maintainer of this file is Richard Heck. If you have comments or error corrections, please send them to the LyX Documentation mailing list, <lyx-docs@lists.lyx.org>.

Contents

1 Introduction **1**

2 LyX and LaTeX **3**
2.1 How LyX Uses LaTeX . . . 3
2.2 Translating LaTeX files into LyX . . . 4
2.3 Inserting TeX Code into LyX Documents . . . 4
2.4 LyX and the LaTeX Preamble . . . 5
2.4.1 About the LaTeX Preamble . . . 5
2.4.2 Changing the Preamble . . . 6
2.4.3 Examples . . . 6
2.4.3.1 Example #1: Offsets . . . 6
2.4.3.2 Example #2: Labels . . . 7
2.4.3.3 Example #3: Paragraph Indentation . . . 7
2.4.3.4 Example #4: This Document . . . 8
2.5 LyX and LaTeX Errors . . . 8

3 Supplemental Tools **11**
3.1 Customizing Bibliographies with BibTeX . . . 11
3.1.1 Alternative Citation Styles . . . 11
3.1.2 Sectioned Bibliographies . . . 11
3.1.3 Multiple Bibliographies . . . 12
3.2 Multipart Documents . . . 12
3.2.1 General Operation . . . 12
3.2.2 Cross-References Between Files . . . 13
3.2.3 Bibliography Lists in all Subdocuments . . . 13
3.3 LyX Archives . . . 14
3.4 Fancy Headers and Footers . . . 14
3.5 Itemize Bullet Selection . . . 15
3.5.1 Introduction . . . 15
3.5.2 How it looks . . . 16
3.5.3 How to use it . . . 16

4 LyX and the World Wide Web **19**
4.1 Math Output in XHTML . . . 20
4.2 Bibliography and Citations . . . 22
4.3 Indexes . . . 22

4.4 Nomenclature and Glossary 23

5 The LyX Server **25**
5.1 Introduction 25
5.2 Starting the LyX Server 25
5.3 Normal communication 26
5.4 Notification 26
5.5 The simple LyX Server Protocol 27
5.6 Reverse DVI/PDF search 27
5.6.1 Automatic setup 27
5.6.2 Manual setup 28
5.6.3 Configuring and using specific viewers 29
5.7 Forward search 31

6 Special Document Classes **33**
6.1 A&A Paper 33
6.1.1 Introduction 33
6.1.2 Getting started 33
6.1.3 The header block 34
6.1.4 The abstract 34
6.1.5 Supported environments 35
6.1.6 Commands not supported by LyX 35
6.1.7 Figure and Table Floats 36
6.1.8 Referee layout 36
6.1.9 The example paper 36
6.2 AASTeX 36
6.2.1 Introduction 37
6.2.2 Starting a New Paper 37
6.2.3 Finishing Your Paper 37
6.2.4 Comments On Specific Commands 38
6.2.4.1 Things that work as expected 38
6.2.4.2 Things that work, but require more comment 38
6.2.4.3 Things not implemented, use TeX code 39
6.2.4.4 Things that cannot be implemented 39
6.2.5 FAQs, Tips, Tricks, and Other Ruminations 40
6.2.5.1 Getting LyX and AASTeX to cooperate 40
6.2.5.2 LaTeX error processing a table 40
6.2.5.3 References 40
6.2.5.4 Including EPS files 40
6.2.5.5 Things I could have done, but didn't 41
6.2.6 Final Caveat 41
6.3 AMS LaTeX 41
6.3.1 What these layouts provide 42

6.4 AGU journals (aguplus) . . . 44
6.4.1 Description . . . 44
6.4.2 New styles . . . 44
6.4.3 New floats . . . 44
6.4.4 Supported journals . . . 45
6.4.5 Bugs and things to remember . . . 45
6.5 Broadway . . . 45
6.5.1 Introduction . . . 45
6.5.2 Special problems . . . 45
6.5.3 Special features . . . 45
6.5.4 Paper size and Margins . . . 45
6.5.5 Environments . . . 46
6.6 Dinbrief . . . 47
6.7 EGS journals (egs) . . . 47
6.7.1 Description . . . 47
6.7.2 New styles . . . 47
6.8 Elsevier Journals . . . 47
6.9 Foils [aka FoilTEX] . . . 47
6.9.1 Introduction . . . 47
6.9.2 Getting Started . . . 48
6.9.2.1 Extra Options . . . 48
6.9.3 Supported Environments . . . 49
6.9.4 Building a Set of Foils . . . 51
6.9.4.1 Give It a Title Page . . . 51
6.9.4.2 Start a New Foil . . . 51
6.9.4.3 Theorems, Lemmas, Proofs and more . . . 51
6.9.4.4 Lists . . . 51
6.9.4.5 Figures and Tables . . . 52
6.9.4.6 Page Headers and Footers . . . 52
6.9.5 Unsupported FoilTEX Goodies . . . 52
6.9.5.1 Lengths . . . 52
6.9.5.2 Headers and Footers . . . 53
6.10 Hollywood (Hollywood spec scripts) . . . 53
6.10.1 Introduction . . . 53
6.10.2 Special problems . . . 54
6.10.3 Special features . . . 54
6.10.4 Paper size and Margins . . . 54
6.10.5 Environments . . . 54
6.10.6 Script jargon . . . 55
6.11 ijmpc and ijmpd . . . 55
6.11.1 Overview . . . 55
6.11.2 Writing a paper . . . 55
6.11.3 Preparing a paper for submission . . . 57
6.11.4 Use of TEX code . . . 57

6.12 iopart . . . 57
6.12.1 Overview . . . 57
6.12.2 Writing a paper . . . 58
6.13 Kluwer . . . 58
6.13.1 Overview . . . 58
6.13.2 Writing a paper . . . 59
6.13.3 Preparing a paper for submission . . . 59
6.13.4 "Peculiarities" of the Kluwer package . . . 59
6.14 Koma-Script . . . 60
6.14.1 Overview . . . 60
6.14.2 article (koma-script), report (koma-script), and book (koma-script) . . . 61
6.14.3 letter (koma-script) . . . 63
6.14.4 The new letter class: letter (koma-script v.2) . . . 65
6.14.5 Problems . . . 66
6.15 Memoir . . . 66
6.15.1 Overview . . . 66
6.15.2 Basic features and restrictions . . . 67
6.15.3 Extra features . . . 68
6.16 The mw Classes . . . 69
6.17 Paper . . . 69
6.18 RevTeX4 . . . 69
6.18.1 Installation . . . 70
6.18.2 Preamble Matter . . . 70
6.18.3 Layouts . . . 70
6.18.4 Important Notes . . . 70
6.18.5 Drawbacks . . . 71
6.19 Springer Journals (svjour) . . . 71
6.19.1 Description . . . 71
6.19.2 New styles . . . 71
6.19.3 Supported journals . . . 72
6.19.4 Credits . . . 72
6.19.5 Bugs . . . 72
6.20 Slides [aka SliTeX] . . . 72
6.20.1 Introduction . . . 72
6.20.2 Getting Started . . . 73
6.20.3 Paragraph Environments . . . 73
6.20.3.1 Supported Environments . . . 73
6.20.3.2 Quirks of the New Environments . . . 75
6.20.4 Making a Presentation with Slide, Overlay and Note . . . 75
6.20.4.1 Using the Slide Environment . . . 75
6.20.4.2 Using Overlay with Slide . . . 76
6.20.4.3 Using Note with Slide . . . 77
6.20.5 The slides Class Template File . . . 78

7 LyX Features needing Extra Software **81**
7.1 Checking TeX . . . 81
7.1.1 Introduction . . . 81
7.1.2 How to use it . . . 82
7.1.3 How to fine tune it . . . 82
7.2 Version Control in LyX . . . 85
7.2.1 Introduction . . . 85
7.2.2 RCS commands in LyX . . . 85
7.2.2.1 Register . . . 85
7.2.2.2 Check In Changes . . . 86
7.2.2.3 Check Out For Edit . . . 86
7.2.2.4 Revert To Repository Version . . . 86
7.2.2.5 Undo Last Checkin . . . 86
7.2.2.6 Show History . . . 86
7.2.2.7 Revision info . . . 86
7.2.3 CVS commands in LyX . . . 87
7.2.3.1 General CVS usage . . . 87
7.2.3.2 Register . . . 87
7.2.3.3 Check In Changes . . . 88
7.2.3.4 Check Out Changes . . . 88
7.2.3.5 Revert To Repository Version . . . 88
7.2.3.6 Update of the local directory checkout from repository 88
7.2.3.7 Show History . . . 89
7.2.4 SVN commands in LyX . . . 89
7.2.4.1 Register . . . 89
7.2.4.2 Check In Changes . . . 90
7.2.4.3 Check Out For Edit . . . 90
7.2.4.4 Revert To Repository Version . . . 90
7.2.4.5 Update of the local directory checkout from repository 90
7.2.4.6 Show History . . . 90
7.2.4.7 File Locking . . . 91
7.2.4.8 Automatic Locking Property . . . 91
7.2.4.9 Revision Information in Documents . . . 92
7.2.5 SVN and Windows Environment . . . 93
7.2.5.1 Preparation . . . 93
7.2.5.2 Bringing a document under Subversion control . . . 93
7.2.5.3 SSH tunnel used with SVN under Windows . . . 94
7.2.5.4 End-of-Line Conversions . . . 94
7.2.6 Further tuning . . . 95
7.2.7 Version control and Document comparison . . . 95
7.3 Literate Programming . . . 95
7.3.1 Introduction . . . 95
7.3.2 Literate Programming . . . 96
7.3.2.1 References . . . 96

7.3.3 LyX and Literate Programming with Noweb 97
7.3.3.1 Generating documents and code (weaving and tangling) 97
7.3.3.2 Configuring LyX 100
7.3.3.3 Debug extensions 101
7.3.3.4 Toolbar extensions 101
7.3.3.5 Colors customization 102
7.3.4 LyX and knitr/Sweave 102

8 Special features **103**
8.1 Multiple Columns 103
8.1.1 Two columns 103
8.1.2 Multiple columns 104
8.1.3 Columns inside columns 105
8.2 Non-standard Paragraph Shapes 106

1 Introduction

This manual is essentially Part II of the *User's Guide.* The reason for separating this document out is simple: the *User's Guide* is already quite lengthy, and it contains information on all of the basic features one needs to know in order to prepare most documents. However, the LyX Team has worked to make LyX extensible through various configuration files and external packages. That means that if you want to support the Fizzwizzle LaTeX package, you can create a layout file (or module) for it without having to alter LyX itself. We've already had contributions of several new features this way, and some of them are documented here. There are also some more 'advanced' features, such as how to control the presentation of bibliographies and how to work with multi-part documents, that are not covered in the *User's Guide* and are discussed here.

This manual also documents some special features, like fax support, version control, and SGML support, which require additional software to work properly. There is also a chapter on LyX's support for HTML. And lastly, there's a chapter of LaTeX tools and tips, things you can use to spruce up your documents by directly using the powerful features of LaTeX. After all, LyX *is* only WYSIWYM and will only ever interface to some, not all, LaTeX features.

If you haven't read the *Introduction* yet, you are definitely in the wrong manual. The *Introduction* is the first place to go, since it describes the notation and format of all of the manuals. You should also be thoroughly familiar with the *User's Guide* and all of the basic features of LyX before attempting to read this one.

Since many of the topics in this manual depend heavily on LyX's interaction with LaTeX, this first chapter covers the inner workings of LyX and how to direct LyX to generate exactly the LaTeX code you want. It is obviously for more seasoned LyX users.

2 LyX and LaTeX

2.1 How LyX Uses LaTeX

This chapter is for both TeX-nicians and the LaTeX-curious. In it, we'll explain how LyX and LaTeX work together to produce printable output. This is the only place in any of the manuals where we assume you know something about LaTeX.

At one time, LyX was called a "WYSIWYM frontend to LaTeX," but that's no longer true. There are frontends to LaTeX out there.[1] These are basically text editors with the ability to run LaTeX and mark any errors in the file you're editing. Although LyX *is* an editor, and it *does* run LaTeX, and it also indicates errors in the file, it also does much, much more. For one thing, you don't need to know LaTeX to use LyX effectively. And LyX has added its own extensions to LaTeX. Try the following sometime: select Export ▷ LaTeX from the File menu (or View ▷ Source), then look at the preamble of the resulting `.tex` file. You'll notice a variety of new macros defined specifically by LyX. These macros are defined automatically, according to the features you use in the document.

There are several commands that automatically invoke LaTeX. They are:

- View ▷ View
- View ▷ Update
- File ▷ Print
- File ▷ Fax

They will only invoke LaTeX if the file has changed since the last time LaTeX was run.

When LyX runs LaTeX on the file you're editing, it performs these steps:

1. Convert the document to LaTeX and save to a file with the extension `.tex` in place of `.lyx`.
2. Run LaTeX on the `.tex` file (maybe several times), and run any other commands (such as `bibtex` or `makeindex`) needed to compile the LaTeX file.
3. If there are any errors, show the error log.

If you've run LaTeX using View ▷ DVI, LyX then runs a DVI viewer to display the DVI-file. If you've used View ▷ PostScript, LyX performs further steps:

[1] Some familar ones are TeXmaker and kile, on Linux, and TeXshop, OSX. There are also the LaTeX modes for vi and emacs, of course.

- Run `dvips` to convert the DVI file to PostScript.
- Run a PostScript viewer, such as `ghostview`, to display the PostScript file.

LyX does similar things when viewing, or exporting, other formats.

2.2 Translating LaTeX files into LyX

You can import a LaTeX file into LyX by using the File ▷ Import ▷ LaTeX command in LyX. This will call a program named `tex2lyx` which will create a file `foo.lyx` from the file `foo.tex`. LyX will then open that file.[2]

`tex2lyx` will translate most legal LaTeX, but not everything. It will put things it doesn't understand into TeX code, so after translating a file with `tex2lyx`, you can look for TeX code and hand-edit it until it looks right.

If you don't know what TeX code is, read the next section.

2.3 Inserting TeX Code into LyX Documents

Anything you can do in LaTeX you can do in LyX, for a very simple reason: You can always insert TeX code into any LyX document. LyX cannot, and will never be able to, display every possible LaTeX construct. If ever you need to insert LaTeX commands into your LyX document, you can use the TeX Code box, which you can insert into your document with Insert ▷ TeX Code.

Here's an example of inserting LaTeX commands in a LyX document. The code looks like this:

```
\begin{tabular}{ll}
\begin{minipage}{5cm}
This is an example for a minipage environment. You
can put nearly everything in it, even (non-floating)
figures and tables.
\end{minipage}
&
\begin{minipage}{5cm}
\begin{verbatim}
\begin{minipage}{5cm}
This ...
\end{minipage}
\end{verbatim}
\end{minipage}
\end{tabular}
```

[2] `tex2lyx` can also be run from the command line, of course.

The TeX Code box containing this text is directly after this paragraph. Those of you reading the manual in LyX will only see the TeX code inset. Those reading a printed version of the manuals will see the actual results:

This is an example for a minipage environment. You can put nearly everything in it, even (non-floating) figures and tables.

```
\begin{minipage}{5cm}
This ...
\end{minipage}
```

In addition to using TeX code, you can also create a separate file containing some complex LaTeX structure and then use Insert ▷ Child Document to include your file (you should select the type `Input`). We recommend that you only do this if you have a `.tex` file which you *know* works already. Otherwise, you'll have a big job tracking down LaTeX errors.

There are a few last points to emphasize:

- LyX *does not* check if your LaTeX code is correct.
- Beware reinventing the wheel.

On that last point, LyX does have quite a few features tucked into it, and more are coming. Be sure to check the manuals to make sure that LyX doesn't have such-and-such feature before you decide you have to do it by hand. Moreover, there are numerous LaTeX packages out there to do all sorts of things, from labels to envelopes to fancy multipage tables. Check out CTAN for details.

If you do need to do some wild and fancy things within your document, be sure to check out a good LaTeX book for assistance. There are a number of them listed in the bibliography of the *User's Guide*.

2.4 LyX and the LaTeX Preamble

2.4.1 About the LaTeX Preamble

If you already know LaTeX, there is no need to explain here what the preamble is good for. If you don't, the following will give you some ideas—we recommend again that you consult a LaTeX book for further information. In any case, you should read the points below, because they explain what you can do and what you don't need to do in the LaTeX preamble of a LyX document.

The LaTeX preamble comes at the very beginning of a document, *before* the text. It serves to:

- Declare the document class.
  LyX already does this for you. If you're a seasoned LaTeX-nician, and you have a custom document class you want to use, check out the *Customization Manual* for information on how to make LyX interface to it.

- Declare the usage of packages.
  LaTeX packages provide special commands, which are only available within a document when the package has been declared in the preamble. For example, the package `indentfirst` forces all paragraphs to be indented. There are other packages for labels, envelopes, margins, etc.

- Set counters, variables, lengths and widths.
  There are several LaTeX counters and variables which *must* be set globally from within the preamble in order to have the desired effect. (There are variables which you can set and reset inside the document, too.) Margins are a good example of something which must be set in the preamble. Another example is the label format for lists. You can actually set these just about anywhere, but it's best to do it just once, inside the preamble.

- Declare user defined commands (with `\newcommand` or `\renewcommand`).
  These are abbreviations for LaTeX commands which appear very often inside a document. Although the preamble is a good place to declare such commands, they *can* be declared anywhere (before they are used for the first time, of course). This can be useful if there is a lot of raw LaTeX code in your document, which normally should not be the case.

LyX adds its own set of definitions to the preamble of the `.tex` file it produces. This makes LaTeX files generated by LyX portable.

2.4.2 Changing the Preamble

The commands which LyX adds to the preamble of a LaTeX file are fixed; you can't change them without patching LyX itself. You can, however, add your own stuff to the preamble by selecting LaTeX Preamble in the Document ▷ Settings dialog. LyX adds anything in the Preamble dialog to its own built-in preamble. Before adding your own declarations in the preamble, you should make sure that LyX doesn't already support what you want to do. (Remember what we said about reinventing the wheel?) Also, *make sure your preamble code is correct.* LyX doesn't check it for you. If there is an error, you're likely to get an error like "`Missing \begin{document}`". If you see this error, check your preamble.

2.4.3 Examples

Here are some examples of what you can add to a preamble, and what they do.

2.4.3.1 Example #1: Offsets

There are two variables under LaTeX that control page position: `\hoffset` and `\voffset`. Their names should be self-explanatory. These variables are useful if you think for a moment about computer labels. Sometimes, the size of a print medium

and the area of the medium that you can actually print on aren't the same. This is where `\hoffset` and `\voffset` come in.

The default values for `\hoffset` and `\voffset` are both 0 points, i. e. the page isn't shifted. Unfortunately, some DVI drivers always seem to shift the page. We have no idea why, or why the sysadmin hasn't fixed such behavior. If you're using LyX on a system that you don't personally maintain, and your sysadmin is a doofus, `\hoffset` and `\voffset` can save the day. Suppose your left and top margins are always 0.5 inches too big. You can add this to the preamble:

```
\setlength{\hoffset}{-0.5 in}
\setlength{\voffset}{-0.5 in}
```

and your margins should now be correct.

2.4.3.2 Example #2: Labels

Speaking of labels, suppose you wanted to print out a bunch of address labels. There's a rather nice package, available at your nearest CTAN archive, for printing sheets of labels: `labels.sty`. Now, your system may not have this package installed by default. We leave that up to you to check. You'll also want to read the documentation for it; we're not going to do that for you. Since this is an example, however, we'll give you an example of how you use this package.

First, make sure you're using the `article` document class. Next, you need to put the following in your preamble:

```
\usepackage{labels}
\LabelCols=3
\LabelRows=7
\LeftBorder=8mm
\RightBorder=8mm
\TopBorder=9mm
\BottomBorder=2mm
```

This sets things up for Avery label sheets, stock #5360. You're now ready to print labels, but you'll need to insert LaTeX code, placing the commands `\begin{labels}` and `\end{labels}` around each label text. This and other special features of `labels.sty` are explained in its documentation.

Someday, someone may write a LyX layout file to support this package directly. Maybe that someone is you.

2.4.3.3 Example #3: Paragraph Indentation

Americans are trained to indent the first line of *every* paragraph. As with all of their other weird quirks, most Americans will whine and moan until they can have their

way and indent the first line of all paragraphs. (Yes, we're joking. (We are?) *Yeah*, we are.)

Of course, this behavior isn't standard typography. In books, you typically only indent the first line of a paragraph *if* it follows another one. The idea behind indenting the first line of a paragraph is to distinguish neighboring paragraphs from one another. If there is no previous paragraph—for example, if it follows a figure or is the first paragraph in a section—then there is no need for indentation.

If you're a typical American (we're still joking!), though, you don't care about such esoteric things; you want your indentation! Add this to the preamble:

```
\usepackage{indentfirst}
```

If your TeX distribution isn't braindead, you'll have this package, and all of your paragraphs will get the indentation the Founding Fathers intended they should have.

2.4.3.4 Example #4: This Document

You can also check out the preamble of this document to get an idea of some of the advanced things you can do.

2.5 LyX and LaTeX Errors

When LyX calls LaTeX, it tells LaTeX to blithely ignore any errors and keep going. It then uses the logfile from the LaTeX run to do a post-mortem. After analyzing the logfile,LyX displays a dialog listing the errors. Clicking on any one of them will take you to the position in your LyX file where the error occurred.[3]

Some folks also like to look at the log file directly: It is available from Document ▷ Latex Log. There are some fairly common error messages and warnings. We'll cover those here. You should look at a good LaTeX book for a complete listing.

- `LaTeX Warning`
  Anything beginning with these words is a warning message for the purpose of "debugging" the LaTeX code itself. You'll get messages like this if you added or changed cross-references or bibliography entries, in which case, LaTeX is trying to tell you that you need to make another run. You can by-and-large ignore these.

- `LaTeX Font Warning`
  Another warning message, this time about fonts which LaTeX couldn't find. The rest of the message will often say something about a replacement font that LaTeX used. You can safely ignore these, too.

[3] Well, usually. Analyzing the logfile is a tough job, and LyX doesn't always go to the right line. There are also cases where LaTeX reports the error on one line, but the actual error is earlier. This is not unlike forgetting a closing brace in a program: You'll get an error, but only later.

- `Overfull \hbox`
  LaTeX absolutely *loves* to spew these out. They are warnings about lines that were too long and run past the right margin. Almost always, this is unnoticeable in the final output. (It can be just a point or two.) Or, only one or two characters extend past the margin. LaTeX seems to generate at least one of these messages for just about any document you write.
  You can ignore these messages. Your eyes will tell you if there's a problem with something that's too wide; just look at the output.[4]

- `Underfull \hbox`
  Not quite as common as its cousin. LaTeX seems to like to print lines that are a bit too wide as opposed to ones that are a bit too narrow. We have no idea why.

- `Overfull \vbox` and `Underfull \vbox`
  Warnings about troubles breaking the page. Once again, just look at the output. Your eyes will tell you where something has gone wrong.

- `LaTeX Error: File 'Xxxx' not found`
  The file "Xxxx" isn't installed on this system. This usually appears because some package your document needs isn't installed. If you didn't touch the preamble or didn't use the `\usepackage{}` command, then one of the packages LyX tried to load is missing. Use Help ▷ LaTeX Configuration to get a list of packages that LyX knows about. This file is updated whenever you reconfigure LyX (using Tools ▷ Reconfigure) and tells you which packages have been detected and what they do.
  If you did use the `\usepackage{}` command and the package in question isn't installed, then you'll need to install it yourself.

- `LaTeX Error: Unknown option`
  Error messages beginning with this are trying to tell you that you specified a bad or undefined option to a package. Check the package's documentation.

- `Undefined control sequence`
  If you've inserted LaTeX code into your document, but made a typo, you'll get one of these. You may have forgotten to load a package. In any case, this error message usually means that you used an undefined command.

There are other error and warning messages. Some are self-explanatory. These are usually LaTeX messages. Others are downright cryptic. These are usually TeX error messages, and we really have *no clue* what they mean or how to decipher them. No-one does.

There's a general sequence you should follow if you get error messages:

[4] You can also enable the 'draft' option in Document ▷ Settings, and then LaTeX will draw a black box in the margin of lines that are overfull.

1. Look at the LaTeX code you inserted for typos.
2. If there are no typos, check that you used the command(s) correctly.
3. If you get a bunch of error boxes piled up at the very top of the document—and especially if you see a "`Missing \begin{document}`" error—it means that there are errors in the preamble. Start debugging your preamble.
4. If you didn't add anything to the preamble and didn't add any LaTeX code to the document, the first suspect is your LaTeX distribution itself. Check for missing packages and install them.
5. Okay, so there are no missing packages. Did you use any of the fine-tuning options in LyX? Specifically, did you *misuse* any of them, like trying to manually insert lots of `Protected Blanks`, `Linebreaks`, or `Pagebreaks`? Did you try to kludge something together with these instead of using the appropriate paragraph environment?
6. All right, you didn't use any of the fine-tuning options, you played by the rules. Did you try to pull a fancy maneuver? Did you do something funky inside a table or an equation, like inserting a graphic into a table cell?
7. Do you have long sections of text where LaTeX cannot find a place to break a line? By default, LaTeX is rather strict about how much extra inter-word spacing it will add in order to break a line. Preferably, you should rework the paragraph to avoid the problem. If this isn't an option, you can wrap your text in `\sloppypar` to make LaTeX's line breaking more, well, sloppy.
8. Did you go overboard with the nesting? LyX (currently) doesn't check to make sure you're in the limits for nesting environments. If you nested a bunch of environments to the 17^{th} level, that's the problem. (The limit in LaTeX is five.)
9. Okay, you didn't get any error messages, but your output looks awful. If you have a table or figure that's too wide or long for the page, you need to:
   a) rescale the figure so it fits.
   b) trim down the table so it fits.
10. If something else is wrong with the output, and you didn't try to pull anything fancy or kludge the fine-tuning options, we're not sure what's wrong.

If all this doesn't help—well, then *perhaps* you might have found a bug in LyX....

3 Supplemental Tools

3.1 Customizing Bibliographies with BibTEX

The most basic information about how to use BibTEX with LYX is contained in the section *Bibliography databases (BibTEX)* of the *User's Guide*. The following subsections explain special bibliography features supported by LYX.

3.1.1 Alternative Citation Styles

Standard BibTEX uses numbers (e. g. "[12]") to refer to a cited work. However, in many scientific disciplines, other citation styles are in use. The most common one is the author-year style (e. g. "Knuth 1984a"). LYX supports two packages that provide this style, `natbib` and `jurabib`. Both packages have their pros and cons, which cannot be listed in detail. If you only want to have simple author-year (or author-numerical) style, or if you want to use one of the countless style files for `natbib`, than the established `natbib` package is probably your choice. If you need special features like short title references, ibidem etc., you might consider the `jurabib` package.

The handling of both packages in LYX is basically the same. Go to Document ▷ Settings and select under Bibliography the option Natbib or Jurabib. With both packages, you will get some extra features in the citation dialog and you can select the style of the reference ("Knuth 1984", "Knuth (1984)", "Knuth, 1984", "1984" etc.). Note that both packages need specifically designed style files. They both ship their own, but there are lots of additional style files, and there is even an interactive style file builder[1] for `natbib`.

3.1.2 Sectioned Bibliographies

Sometimes you might need to divide your bibliography into several sections. If you are for instance a historian, the possibility to separate sources and scientific works is most likely a "must have". Unfortunately, BibTEX itself does not allow you to do this. But with the help of some LATEX packages, BibTEX can be extended to fit your needs.

LYX provides native support for one of these packages, `bibtopic`.[2] The advantage of this package (compared to other packages like `multibib`) is that you don't need to define new citation commands. Instead, you need to prepare different bibliographic

[1] See ftp://ctan.tug.org/tex-archive/macros/latex/contrib/custom-bib/

[2] Available from ftp://ctan.tug.org/tex-archive/macros/latex/contrib/bibtopic/

databases which include the entries for the different sections of the bibliography. For example: If you want to divide your bibliography into the sections "Sources" and "Scientific works", you first need to create two bibliographic databases, e. g. `sources.bib` and `scientific.bib`.

Go to Document ▷ Settings and check under Bibliography the option Sectioned bibliography. Now you can insert multiple BibTEX bibliographies, one for each section of your bibliography. Returning to our example: Insert the BibTEX bibliography `sources.bib` and a second one for the database `scientific.bib`. You are free to use the same or different styles for each section. Additionally, you can chose if the bibliography section should contain "all cited references" of the specified database(s) (which is the default), "all uncited references" or even "all references". This might be useful if you would like to separate your bibliography into three sections: "Cited sources", "Uncited sources", and "Scientific works". The titles for the sections can be added as ordinary sections or subsections. Since `bibtopic` removes the bibliography title, you have manually re-add that, too (as a chapter* or section*, for instance).

3.1.3 Multiple Bibliographies

Multiple bibliographies, e. g. a bibliography for each section or chapter of the document, are not supported by BibTEX itself. But the `bibtopic` package, which is used for the creation of sectioned bibliographies in LyX (see the previous section), provides an easy way to solve this task, if you are willing to use some TEX Code (see section 2.3).[3]

First go to Document ▷ Settings and under Bibliography check Sectioned bibliography. In the document, you have to enclose the sections, which shall contain their own bibliography (including the BibTEX bibliography itself), between `\begin{btUnit}` and `\end{btUnit}` (those commands have to be inserted as TEX code). The bibliography will contain all references which have been cited in the current `btUnit`. **Note:** If you are using this approach, then every citation reference has to be inside some `btUnit`. Also, the `btUnit`s cannot be nested.

3.2 Multipart Documents

3.2.1 General Operation

When you are working on a large file with many sections, it is often convenient to break up the document into several files, or perhaps you have something where a table may change from time to time, but the preceding text does not. In these cases, you should seriously consider using multipart documents. For example, scientific papers often have five major sections: the introduction, observations, results, discussion, and conclusion. Each of these could be its own separate LyX file, with one "master" file which contains the title, authors, abstract, references, etc., plus the five included files.

[3] An alternative approach is to use the `chapterbib` or `bibunits` package, respectively.

It is important to note that each of these files is a full LyX file which can be formatted and printed on its own, as well as included in a master file. Each of these files must have the same document class, however—don't attempt to mix book classes with article classes. You may also include LaTeX files; however, these files must not have their own preamble (i. g. everything up to and including the `\begin{document}` line as well as the `\end{document}` line must be deleted) or else errors will be generated when you try to make a DVI file.

LyX allows you to include files quite easily with Insert ▷ Child Document. When you click on this selection a small box is inserted into the file at the current cursor location. Clicking on the box raises a dialog which allows you to select the file to be included, and the method of its inclusion.

The file selection box should by now be obvious. The three inclusion methods are "include", "input", and "verbatim". The difference between "include" and "input" is really only meaningful to LaTeXperts, but the practical difference is that files which are "included" are typeset beginning on a new page, while files which are "inputted" are typeset starting on the current page.

Generally, the master file is converted into a full LaTeX file before typesetting, while the included files are converted to LaTeX files which do not have all the preamble information.

A "verbatim" included file allows you to include a file typeset exactly as it appears in the file, i. g. in `verbatim` mode, with the characters set in a fixed-width typewriter font. Normally, spaces in this file are invisible, though two consecutive spaces are conserved, unlike LyX's normal treatment of spaces. However, setting the Mark spaces in output checkbox typesets a mark to unambiguously define the presence of a space.

3.2.2 Cross-References Between Files

This section is somewhat out of date. Need to describe default master documents and how children are opened when the master is. [[FIXME]]

It is possible to set up cross-references between the different files. First, open all the files in question: let's call them A and B in a two file example, where B is included in A. Let's say you insert a label in A, then want to reference it in B. Open the cross-reference dialog in whilst in document B, and you can select the "buffer" to use.

3.2.3 Bibliography Lists in all Subdocuments

This section also needs updating. There is now material about this on the wiki, and it could be copied here.

Copy the bibliography list with all entries to all subdocuments and transform them to a comment. This way LyX will find the `.bib`-files and you can easily insert references without making the bibliography list visible.

As the bibliography list is in a comment, LaTeX won't use it and the references will look like this: [?], instead of like this: [1]. One solution is to use the LaTeX-package `comment` that will only include comments by processing the files separately. To do this, add in the LaTeX preamble of every subdocument the following:

```
\usepackage{comment}
\includecomment{comment}
```

See also http://wiki.lyx.org/FAQ/Unsorted#toc31.

3.3 LyX Archives

Users sometimes need to be able to "bundle" a LyX file together with all the images (and other files) on which it depends, either for sending to a publisher or for sharing with a co-author. LyX 2.0 includes a Python script (`lyxpak.py`) that automates this process. To use it, you must have either the `zipfile` or `tarfile` python modules installed on your system. By default, the script prefers the `gzip`-compressed `tar` format on Unix-like systems and the `zip` format on Windows.

LyX's configuration process will set the script up to export a 'LyX Archive', and this format will then be available under `File` ▷ `Export`.

Independently of the platform, the generation of a particular archive format can be forced by adding either the `-t` (for the `tar` format) or `-z` (for the `zip` format) switch to the LyX ▷ `LyX Archive` converter in `Tools` ▷ `Preferences` ▷ `File Handling` ▷ `Converters`. (Make sure you add the switch after the script name, not before it.)

3.4 Fancy Headers and Footers

The default page layout is rather plain; for an `article` document class, all you get is a centered page number at the bottom of the page. This document uses KOMA-script's book class, so it appears to be a bit fancier. But to really put on a show, you need to set the document page style to "fancy", as mentioned in the *User Guide*. This section describes the LaTeX code you need to insert in your LaTeX preamble in order to get the desired effects.

For more information on fancy headers, see the `fancyhdr` package documentation. (http://www.ctan.org/tex-archive/help/Catalogue/entries/fancyhdr.html)

The page header is divided into three fields, not surprisingly labeled "left", "center", and "right". The footer is also divided into these three fields. The LaTeX commands to set these fields in the simplest manner are `\lhead`, `\chead`, `\rhead`, `\lfoot`, etc. Suppose you wish to put your name in the upper left hand corner of each page. Simply insert the following command in the preamble:

```
\lhead{John Q. DocWriter}
```

You will now see your name in the upper left. If a field has a default entry that you would like to get rid of (often the page number appears in the central footer, simply include a command with a blank argument, e. g.:

```
\cfoot{}
```

Let's get really fancy: lets put the section number with the word "Section" (e. g. Section 3) in the upper left, the page number (e. g. Page 4) in the upper right, your name in the lower left, and the date in the lower right. The following commands should now appear in the preamble:

```
\lhead{Section \thesection}
\chead{}
\rhead{Page \thepage}
\lfoot{John Q. DocWriter}
\cfoot{}
\rfoot{\today}
```

The commands `\thesection` and `\thepage` access LaTeX's section and page counters, and so print out the current section and page numbers. `\today` simply prints out today's date.

The thicknesses of the horizontal rules drawn beneath the header and above the footer can also be modified. If you don't want one of the rules, set its thickness to 0. The header rule has a default thickness of 0.4pt, the footer rule is 0pt. Use commands like `\renewcommand{\headrulewidth}{0.4pt}` and `\renewcommand{\footrulewidth}{0.4pt}` to set the thicknesses.

You can switch the header/footer settings on and off for individual pages using commands like `\thispagestyle{empty}`, `\thispagestyle{plain}`, and `\thispagestyle{fancy}`. Simply insert them in the text on the page you want changed and mark them as TeX code. In fact, title pages are marked as plain by default, while following pages are marked fancy when using the global fancy setting.

As a final example, it is possible to include an image in the header or footer. Suppose you want to put a company logo in the upper lefthand corner. You might try something like

```
\lhead{\resizebox{1in}{!}{\includegraphics{logo.eps}}}
```

(you may need to preface this with `\usepackage{graphics}` if you don't include graphics elsewhere in your document).

3.5 Itemize Bullet Selection

by Allan Rae

3.5.1 Introduction

LyX provides 216 bullet shapes that can be accessed from a simple dialog. Using this dialog you can easily specify what bullet shape to use at each level of an itemized list. These settings are document-wide so you won't be able to specify different sets of bullets for different paragraphs.[4]

[4] Well, actually you can but you'll have to do it by hand.

3.5.2 How it looks

Open the dialog by selecting the Document ▷ Settings menu item and then select the Bullets tab.

The dialog provides you with a table of bullet shapes. A column of buttons on the left of the table provides access to the six different panels of bullet shapes. The row of buttons across the top is used to select which bullet depth you are changing. A text entry under the table shows the currently selected bullet shape's LaTeX equivalent and this can be edited if desired. If you do modify the text you will also need to specify any needed packages in the LaTeX preamble.

The six panels are divided up by the packages they require. The following table shows the mappings from button name to LaTeX packages.

Button	Packages Required
Standard	base LaTeX
Maths	`amssymb.sty`
Ding1	`pifont.sty`
Ding2	`pifont.sty`
Ding3	`pifont.sty`
Ding4	`pifont.sty`

LyX doesn't stop you using bullets from packages you don't have. If you get errors from LaTeX when you try to view or print the file, then it is likely you are missing a package.[5]

3.5.3 How to use it

Select which bullet depth you want to change then select the bullet shape and size. Any changes will not be visible in LyX, but are visible when viewing the document.

You can reset a bullet shape to the default simply by clicking your right mouse button on the appropriate bullet depth button.[6]

[5] LyX doesn't restrict your use since you may be editing locally and exporting elsewhere.

[6] If you *really* want to have multiple sets of paragraphs with different sets of bullets in each, then you're going to have to get your hands dirty with TeX code. The bullet selection dialog can help though because it provides you with the LaTeX code for a wide range of bullet shapes. To make your own custom paragraphs you have the following options:

- Use the LaTeX command `\renewcommand{}{}` to specify a new bullet shape for a given depth. You'll also need to save the current bullet shape so you can restore it again afterwards. In this itemized list the following LaTeX code was used to change the bullet used for the first depth.

```
\let\savelabelitemi=\labelitemi
\renewcommand\labelitemi[0]{\small\(\sharp\)}
```

- Note that the itemize depth is specified in Roman numerals as part of the `\labelitem` command.

⋆ Specify each individual entry by starting each item with the bullet shape enclosed in square brackets and set as TEX Code. For example, this item was started with `[\(\star\)]`.

You'll also need to revert the labelitem back to its previous setting for the global bullet shape settings to remain in effect. The way used here was:

```
\renewcommand\labelitemi[0]{\savelabelitemi}
```

4 LyX and the World Wide Web

LyX has long supported the export of documents to various web-friendly formats, such as HTML. Before verson 2.0, however, HTML export was always accomplished by the use of external converters.[1] These fall into two large groups: there are converters that use LaTeX as an intermediate format, such as `htlatex`, `html2latex`, and `plastex`, relying upon LyX to produce the LaTeX; and there is a converter that works directly on LyX files, `eLyXer`. All of these have different advantages and disadvantages.

The LaTeX-based converters have the advantage that, in principle, they know everything about the exported document that LaTeX does. Such converters do not care, for example, if a certain block of code was produced by LyX or was Evil Red Text. These converters know about `aux` files, counters, and references, and can often make use of the `bbl` files generated by BibTeX. On the other hand, LaTeX is a very hard language to parse—it is sometimes said that only TeX itself understands TeX—and LaTeX-based converters will often choke on what seem to be quite simple constructs; complex ones can throw them completely, and as of this writing, for example, the *Embedded Objects* manual will not export with `htlatex`, though the others will. The other issue concerns how math is handled. These converters typically convert the formulae into little pictures that are then linked from the HTML document. Since these are actually generated by LaTeX, they are accurate. But they do not scale well, and just getting them to look as if they are actually meant to be in your document—so that the font sizes seem to be roughly the same—can be a challenge.

Alex Fernandez's `eLyXer` (http://elyxer.nongnu.org/) solves some of these problems. In particular, the formulae it produces scale perfectly, since it renders math using a combination of HTML and CSS rather than converting the formulae to images. But even moderately complex formula are rendered less well than with `htlatex`; this reflects the limitations of HTML.[2] More importantly (as of version 1.1.1, at least), `eLyXer` has limited support for math macros and no support for user-defined paragraph or character styles. These limitations make `eLyXer` unsuitable for many of the documents LyX users produce. In principle, of course, these problems could be solved, but the LyX developers have decided to follow a different path and have made LyX itself capable of writing XHTML, just as it is capable of writing LaTeX, DocBook, and plaintext.

As of this writing, XHTML output remains under development and should prob-

[1] For details on the use of external converters with LyX, see the *Converters* section of the Customization manual.

[2] That said, `eLyXer` can also use jsMath and MathJax for equations, but this setting is only available globally and requires one to have access to a server that runs the backend.

ably be regarded as "experimental".[3] Still, the developers have chosen this approach because it has several potential advantages over the other two.

These advantages are primarily due to the fact that the XHTML output routines, since they are part of LyX, know everything LyX knows about the document being exported.[4] So they know about the table of contents (as displayed in the outline), about the counters associated with different paragraph styles, and about user-defined styles. The XHTML output routines know what LyX knows about internationalization, too, so they will output "Chapter 1" or "Kapitel 1", depending upon the language in effect at the time.

Quite generally, the output routines know what LyX knows about document layout, that is, about how the document is to be rendered on screen. We use this information when we output the document as XHTML. In particular, LyX *automatically* generates CSS style information corresponding to the layout information it uses to render the document on screen: if section headings are supposed to be sans-serif and bold as seen in LyX, then (by default) they will be sans-serif and bold when viewed in a web browser, too. And this is true not just for pre-defined styles, like Section, but for any style, including user-defined styles. Indeed, the XHTML output routines make no distinction between user-defined paragraph and character styles and LyX's own pre-defined styles: in each case, everything LyX knows about the styles is contained in the layout files. And much the same is true as regards pre-defined textual insets, such as footnotes, and various custom insets.

The result is that XHTML output can be customized and extended in exactly the same way LaTeX output can be customized and extended: through layout files and modules. See chapter five of the *Customization* manual for the details.

The remainder of this chapter contains more detailed information on XHTML output, its limitations, and ways to work around those limitations.

4.1 Math Output in XHTML

LyX offers four choices for how math is rendered. These have various advantages and disadvantages:

- MathML
  MathML is a dialect of XML designed specifically for mathematics on the web, and it typically renders very well in browsers that support it. The disadvantage is that not all browsers support MathML, and support is not complete even in the Gecko-based browsers, such as Firefox.
  If LyX is unable to render a formula as MathML—for example, if the formula

[3] The file `development/HTML.notes`, which can be found in the LyX source tree or accessed online, usually contains up-to-date information about the state of XHTML output. See also the list of XHTML bugs on the bug tracker.

[4] Another advantage is that, since these routines are internal to LyX, they are immune to changes in LyX's file format, or to changes in the semantics of existing insets.

uses the `xymatrix` package or ERT—then it will instead output the formula as an image.

- HTML
  As mentioned above, `elyxer` outputs math as HTML, styled by CSS.[5] For simple formulae, this can work quite well, though with more complicated formulae it tends to break down. Still, this method has the advantage that it is very widely supported and so it may be appropriate for documents that contain only a little, fairly simple math.
  If Lyx is unable to render a formula as HTML—for example, if the formula uses the `xymatrix` package or ERT—then it will instead output the formula as an image.

- Images
  Like `htlatex`, LyX will output formulae as images, the very same images, in fact, that are used for instant preview.[6] The advantage to this method is that the images are simply generated by LaTeX, so they are very accurate. The disadvantage, as mentioned earlier, is that these are bitmapped images, so they do not scale terribly well, and one cannot copy them, etc. (As of this writing, there are also problems about equation numbering. See bug 6894.)
  The size of the images can be controlled by setting the "Math Images Scaling" parameter under Document ▷ Settings ▷ Output.[7]
  If LyX for some reason fails to create an image for a formula (e.g., if a required LaTeX package is not installed), then it will fall back to outputing the raw LaTeX.

- LaTeX
  Finally, LyX will happily output math as LaTeX. As well as being the output of last resort, this method can be used with such tools as jsMath, which uses JavaScript to render LaTeX embedded in HTML documents. LyX wraps the LaTeX in either a `span` (for inline formulas) or `div` (for displayed formulas) with `class='math'`, as is required for jsMath.

One of these output methods must be selected under Document ▷ Settings ▷ Output. By default, LyX outputs MathML. This is a document-wide setting, therefore.

Eventually, LyX will offer the user the option to select an alternate output method for a particular inset, say, one that isn't being rendered very well by MathML.[8]

[5] LyX has borrowed some of the CSS for its HTML output from `elyxer`.
[6] Instant preview does *not* have to be on for images to be output, however.
[7] For those who want to know, this controls the resolution of the image in dots per inch and is based upon a default of 75 dpi.
[8] That said, since LyX falls back to images if the inset contains ERT, then one can force output as an image by putting some harmless ERT into the math, for example: $a = b$.

4.2 Bibliography and Citations

XHTML output fully supports bibliographies and citations.

Citation labels are generated by the same machinery that generates LyX's on-screen labels, so the labels will look in the output much as they do in LyX, though better. If you are using numerical citations, then LyX will output numerical labels, such as [1] or [17], rather than simply showing the citation key in square brackets, as it does on-screen. If you are using author-year citations, then LyX will add lowercase letters to the years, just as BibTeX does, if it finds more than one citation for a given author-year combination. The labels will be printed with the bibliography entries. Note that there is, at present, no way to customize the appearance of the labels, for example, to choose between square brackets and parentheses.

Bibliography output is handled by the same machinery that handles the presentation of reference information in the citation dialog, so you will see in the XHTML output pretty much what you would see if you were to look at a given entry in the citation dialog. The formatting can be customized in your layout file or, preferably, in a module. See the *Customization* manual for the details.

The main defect at present is that cross-referenced information is printed with every entry with which it is associated. So you can see things like this:

> Jason Stanley, "Context and Logical Form", in *Language in Context: Selected Essays* (Oxford: Oxford University Press, 2007), pp. 30–68.
>
> Jason Stanley, "Semantics in Context", in *Language in Context: Selected Essays* (Oxford: Oxford University Press, 2007), pp. 201–30.

This should be fixed before long.

There is no support at present for sectioned bibliographies. If you have multiple bibliographies, then LyX will print the same bibliography over and over.

4.3 Indexes

LyX will happily export indexes as XHTML, but with certain limitations at the moment.

Index export will be most reliable when you do not attempt to use the fancy constructs that are described in the section on indexes in the *User's Guide*.[9] We'll describe how they are handled using the subsection headings from that section.

- Grouping Index Entries (aka, sub-entries): LyX makes an effort to support these, but the entries must be separated by " ! ", that is, there must be spaces around the exclamation point. This is because it is otherwise too difficult to check for escaped exclamation points, ones in math, and so forth.

[9] The main issue here is that LyX itself does not really handle these. It just lets you enter what you would have to enter in raw LaTeX.

- Page Ranges: There is no support at all for page ranges, since these make no sense with XHMTL. Instead, you will just get two index entries, one at either end.
- Cross-referencing: There is no support for cross-referencing. If LyX finds an entry containing the "|see..." construct, it is just dropped, and the rest is treated as an ordinary index entry.
- Index Entry Order: LyX does support attempts to fix the sorting order. It will take what is before the first '@' it finds and use that for sorting, taking what follows the first '@' to be the actual entry. At present, LyX does not check for escaped '@', so do not try to index email addresses.
- Index Entry Layout: You can format entries by using the text style dialog, or by using any other method available within LyX itself. There is no support for constructs like: "entry|textbf". Indeed, if LyX finds a pipe symbol, '|', in an entry, it will delete it and everything that follows it.
- Multiple Indexes: There is no support for multiple indexes. Rather, all index entries will be printed as one large index. To avoid our printing several versions of the index, we print only the main index, so make sure you have one.

4.4 Nomenclature and Glossary

There is at present no support for glossaries. Adding it would be fairly trivial, and welcome.

5 The LyX Server

5.1 Introduction

The 'LyX server' allows other programs to talk to LyX, invoke LyX commands, and retrieve information about the LyX internal state. This is only intended for advanced users, but they should find it useful. It is by writing to the LyX server, for example, that bibliography managers, such as JabRef, are able to "push" citations to LyX.

5.2 Starting the LyX Server

The LyX server works through the use of a pair of named pipes. These are usually located in `UserDir`, (except on Windows, where *local* named pipes are special objects located in `\\.\pipe`) and have the names "`lyxpipe.in`" and "`lyxpipe.out`". External programs write into `lyxpipe.in` and read back data from `lyxpipe.out`. The stem of the pipe names can be defined in the Tools ▷ Preferences dialog, for example `"/home/myhome/lyxpipe"`, or `"\\.\pipe\lyxpipe"` on Windows (but any working path can be used). You *must* configure this manually in order for the server to start.

LyX will add the '`.in`' and '`.out`' to create the pipes. If one of the pipes already exists, LyX will assume that another LyX process is already running and will not start the server. On POSIX (Unix like) systems, if for some other reason, an unused "stale" pipe is left in existence when LyX closes, then LyX will try to delete it. If this fails for some reason, you will need to delete the pipes manually and then restart LyX. On Windows, pipes are deleted by the OS on program termination or crash, so "stale" pipes should not be possible.

To have several LyX processes with servers at the same time, you have to use different configurations, perhaps by using separate user directories, each with its own `preferences` file, for each process.

If you are developing a client program, you might find it useful to enable debugging information from the LyX server. Do this by starting LyX as `lyx -dbg lyxserver`.

You can find a complete example client written in C in the source distribution as `development/lyxserver/server_monitor.c`.

Another useful tool is command-line based client you will find in `src/client/lyxclient`.

5.3 Normal communication

To issue a LyX call, the client writes a line of ASCII text into the input pipe. This line has the following format:

LYXCMD:*clientname*:*function*:*argument*

clientname is a name that the client can choose arbitrarily. Its only use is that LyX will echo it if it sends an answer—so a client can dispatch results from different requesters.

function is the function you want LyX to perform. It is the same as the commands you'd use in the minibuffer.

argument is an optional argument which is meaningful only to some functions (for instance, the "self-insert" LFUN will insert the argument as text at the cursor position).

The answer from LyX will arrive in the output pipe and be of the form

INFO:*clientname*:*function*:*data*

where *clientname* and *function* are just echoed from the command request, while *data* is more or less useful information filled according to how the command execution worked out. Some commands, such as "font-state", will return information about the internal state of LyX, while other will return an empty data-response. This means that the command execution went fine.

In case of errors, the response from LyX will have this form

ERROR:*clientname*:*function*:*error message*

where the *error message* should contain an explanation of why the command failed.

Examples:

```
echo "LYXCMD:test:beginning-of-buffer:" >~/.lyxpipe.in
echo "LYXCMD:test:get-xy:" >~/.lyxpipe.in
read a <~/.lyxpipe.out
echo $a
```

5.4 Notification

LyX can notify clients of events going on asynchronously. Currently it will only do this if the user binds a key sequence with the function "notify". The format of the string LyX sends is as follows:

`NOTIFY:`*key-sequence*

where *key-sequence* is the printed representation of the key sequence that was actually typed by the user.

This mechanism can be used to extend LyX's command set and implement macros. Bind some key sequence to "notify". Then start a client that listens on the output pipe, dispatches the command according to the sequence, and starts a function that may use LyX calls and LyX requests to issue a command or a series of commands to LyX.

5.5 The simple LyX Server Protocol

LyX implements a simple protocol that can be used for session management. All messages are of the form

LYXSRV:*clientname*:*protocol message*

where *protocol message* can be "hello" or "bye". If "hello" is received from a client, LyX will report back to inform the client that it's listening to it's messages, while "bye" sent from LyX will inform clients that LyX is closing.

5.6 Reverse DVI/PDF search

Some DVI/PDF viewers[1] provide *reverse search* facility (also called *inverse search*). This means that you can tell LyX to put the cursor to a specific line in the document by clicking at the respective position in the DVI/PDF output. To achieve this, the viewer must be able to communicate with LyX. This is done via the LyX server either by using the named pipe (*lyxpipe*), or the UNIX domain socket (*lyxsocket*) that LyX creates in its temporary directory (this is the way the `lyxclient` program communicates with LyX). In some cases, you need a helper script that mediates between the viewer and LyX, in others, the viewer can communicate with LyX directly. This depends on the selected viewer and on your operating system. The same applies to the way viewers need to be configured and the way the reverse search is actually performed. In what follows, we will thus describe how to setup reverse search for specific viewers. Before we turn to this, though, we will explain what needs to be done generally to enable reverse search in the DVI/PDF output.

5.6.1 Automatic setup

In most cases LyX will do the work for you by enabling the option Synchronize with Output in the menu Document ▷ Settings ▷ Output. In such a case LyX will automatically insert the necessary SyncTeX macro (for PDF) or load the srcltx package (for DVI) respectively. This option can be easily reached also on `View/Update` Toolbar.

[1] The following viewers offer the reverse PDF search feature: Okular on KDE/Linux, Skim on Mac OSX and SumatraPDF on Windows.

If you need a different setting, you can select or input a custom macro from the drop down box just below Document ▷ Settings ▷ Output ▷ Synchronize with Output.

Note that the method `\synctex=1` enables gzip compression. If your viewer does not support it, you should instead use `\synctex=-1`.

Please aöso note that including the `srcltx` package or `src-specials` sometimes has an undesired impact on the typesetting. Thus, you should switch output synchronization off for the final typesetting if you use the `srcltx` package or `src-specials`.

In case you need some more special settings that are not covered by the automatic settings, read the next section about how to set up output synchronization manually. If the automatic setup suits your needs, you can readily jump to section 5.6.3, where the necessary configuration steps in your viewer – needed both with the automatic and the manual setup – are described.

5.6.2 Manual setup

LaTeX provides several different methods for reverse search. Some are built-in in the `latex/pdflatex` program, some are provided by external packages. Your choice depends on whether your LaTeX distribution already provides a given method (the built-in methods are rather new) and whether your viewer can cope with it. The available methods are described in the following.

Built-in DVI-search via src-specials (DVI only)

This method provides the DVI file with the necessary information for reverse search. It is available in LaTeX since quite some time (any somewhat recent LaTeX distribution should include it), and it works reliably. To enable it, change the LaTeX (plain)->DVI or LaTeX (plain)->DraftDVI converter in Preferences ▷ File Handling ▷ Converters to `latex -src-specials $$i`. If this doesn't work, check if your TeX engine needs different options (the syntax might differ in some distributions).

External Packages (PDFSync and scrltx)

The packages *pdfsync* and *scrltx* provide reverse search facility for PDF output (via pdflatex) and DVI output, respectively. In order to enable it, load the packages in the LyX preamble:

- `\usepackage{pdfsync}` for reverse PDF search,
- `\usepackage[active]{srcltx}` for reverse DVI search.

If you want to be able to perform both DVI and PDF reverse searches, you can also insert in the preamble the following lines

```
\usepackage{ifpdf}
\ifpdf
  \usepackage{pdfsync}
```

```
\else
  \usepackage[active]{srcltx}
\fi
```

This way, you can preview the file as either DVI or PDF (pdflatex) and the right package will be used.

Note that PDFSync might affect the output layout of your document. It is therefore advised to disable PDFsync for final documents.

Built-in reverse search via SyncTEX (DVI and PDF)

Recent versions of `(pdf)latex` have built-in support for both PDF and DVI reverse search. This so-called *SyncTEX* facility is basically the result of the integration of the PDFSync package to the pdftex program and its merge with the *scr-specials* approach. You need at least TEXLive 2008 or a recent MikTEX distribution in order to use it. Also note that only a few PDF viewers (Skim on the Mac, SumatraPDF on Windows) already provide SyncTEX support.

To enable SyncTEX for DVI output, change the LaTeX (plain) -> DVI or LaTeX (plain) -> DraftDVI converter in Preferences ▷ File Handling ▷ Converters to `latex -synctex=1 $$i`, and for PDF output, change the LaTeX (pdflatex) -> PDF (pdflatex) or converter to `pdflatex -synctex=1 $$i`. Check the documentation of your viewer whether the viewer needs to be configured for the use with SyncTEX.[2]

5.6.3 Configuring and using specific viewers

Xdvi (all platforms)

If you use `xdvi`, you don't need to do anything else for performing a reverse DVI search, as LyX already provides the necessary hooks for automatically using the `lyxclient` program. Just setup your document as described above (reverse search is triggered by Ctrl-click or Alt-click on Mac OSX, respectively).

However, if for whatever reason you want to use the named pipe instead of the socket for communicating with LyX, simply change the DVI viewer in Preferences ▷ File Handling ▷ File formats to[3] `xdvi -editor 'lyxeditor.sh %f %l'`, where `lyxeditor.sh` is a suitable script. For example, a minimal shell script is the following one:

```
#!/bin/sh
LYXPIPE="/path/to/lyxpipe"
COMMAND="LYXCMD:revdvi:server-goto-file-row:$1 $2"
echo "$COMMAND" > "${LYXPIPE}".in || exit
read < "${LYXPIPE}".out || exit
```

[2]Note that the option `-synctex=1` option enables gzip compression. If your viewer does not support it, you should instead use `-synctex=-1`.

[3]On Mac OSX you have to use `DISPLAY=:0.0 xdvi -editor 'lyxeditor.sh %f %l'`

where `/path/to/lyxpipe` is the LyXServer pipe path specified in Preferences ▷ Paths.[4]

MacDviX (Mac OSX)

At the end of `/Applications/MacDviX_Folder/calleditor.script`, add the following lines:

```
/Applications/LyX.app/Contents/MacOS/lyxeditor "$2" $1
exit 1
```

Modify the lines accordingly if you install LyX somewhere else than in the Applications folder.

Reverse search is triggered by Alt-click (OPTION-click).

Skim (Mac OSX)

Enter `open -a Skim.app $$i` to the viewer setting in Preferences ▷ File Handling ▷ File formats ▷ PDF (pdflatex), and then in Skim ▷ Preferences ▷ Sync choose custom preset and enter command /Applications/LyX.app/Contents/MacOS/lyxeditor.

Reverse search is triggered by COMMAND-SHIFT-click

Okular (KDE)

Go to Settings ▷ Configure Okular... ▷ Editor, select "Custom Text Editor" and add the command `lyxclient -g %f %l`.

Reverse search is triggered by SHIFT-click.

YAP (Windows)

Launch yap, choose its View ▷ Options menu and select the "Inverse DVI Search" tab. Click on the "New..." button and, in the window that opens, enter "LyX Editor" (or any other name you like) in the "Name:" field. Now click on the button labeled "..." to open a file dialog and navigate to the directory containing the batch file `lyxeditor.bat` (see below). Select `lyxeditor.bat` and then specify the program arguments as `%f %l`. The `lyxeditor.bat` wrapper is used for communicating with LyX through the *lyxpipe* and is as follows:

```
@echo off
echo LYXCMD:revdvi:server-goto-file-row:%1 %2> \\.\pipe\lyxpipe.in
type \\.\pipe\lyxpipe.out
```

Make sure that the LyXServer pipe path you specified in LyX is `\\.\pipe\lyxpipe`, otherwise change the `lyxeditor.bat` wrapper accordingly.

In yap, reverse search is triggered by double-click.

[4] In the `development/tools` folder of a source distribution you can find a `lyxeditor` script which is able to locate the *lyxpipe* based on your preferences.

SumatraPDF (Windows)

In order to use SumatraPDF for inverse search, enter `SumatraPDF -inverse-search "lyxeditor.bat %f %l"` in the viewer setting in Preferences ▷ File Handling ▷ File formats ▷ PDF (pdflatex), where `lyxeditor.bat` is the previous wrapper. If SumatraPDF.exe is not in your command PATH, use its full file name.

Reverse search is triggered by double-click.

YAP (Cygwin)

First of all, make sure that yap is your default DVI viewer in the Windows environment, then launch it, choose its View ▷ Options menu and select the "Inverse DVI Search" tab. Click on the "New..." button and, in the window that opens, enter "LyX Editor" (or any other name you like) in the "Name:" field. Now click on the button labeled "..." to open a file dialog and navigate to the directory containing the `lyxeditor.exe` program (which is installed by default on Cygwin along with the LyX executable). Select `lyxeditor.exe` and then specify the program arguments as `-g %f %l`. In this way, you will be using the *lyxsocket* for communicating with LyX. If, for whatever reason, you want to use the *lyxpipe*, omit the `-g` option and be sure to specify the LyXServer pipe path in the LyX preferences.

In yap, reverse search is triggered by double-click.

SumatraPDF (Cygwin)

In order to use SumatraPDF for inverse search, enter `SumatraPDF -inverse-search "lyxeditor -g %f %l"` in the viewer setting in Preferences ▷ File Handling ▷ File formats ▷ PDF (pdflatex). If SumatraPDF.exe is not in your command PATH, use its full posix path. The `-g` enables communication via the *lyxsocket*. Again, omit the `-g` option if you want to use the *lyxpipe*, and be sure to specify the LyXServer pipe path in the LyX preferences.

Reverse search is triggered by double-click.

5.7 Forward search

Forward search is, as the name implies, in a sense the "opposite" to reverse search. It allows you to let the viewer jump to a given position from within LyX. If forward search is set up (as described in what follows), you can put the cursor anywhere in your LyX document, and hit Navigate ▷ Forward search (or select Forward search in the context menu via right mouse click), and then the viewer will jump to that position as well. This implies, of course, that your viewer supports this function.

To make forward search possible, you first need to provide the generated output PDF/DVI file with additional information about the TeX sources. This can be done via the methods described in the section 5.6.1.

Additionally, you need to configure LyX for using your viewers in Tools ▷ Preferences ▷ Output ▷ General. We provide a range of tested configurations for some viewers, which you can select from the drop down list. If none of these configurations suits you, you have to find out and enter a suitable configuration yourself. The definition syntax uses the following placeholders:

- `$$n`: row number
- `$$t`: name of the (temporary) exported .tex file (without path)
- `$$f`: name of the (temporary) exported .tex file (including path)
- `$$o`: name of the exported output file (either dvi or pdf, depending on which one exists in the temporary directory)

Note that only some of the viewers provide full forward search functionality, among them yap, xdvi, okular and Sumatra PDF.[5] While many of the widespread PDF viewers (most notably Adobe Reader) do not support forward search at all, some other viewers – e. g. evince and xpdf – allow at least to reload the document and jump to a specific page of the file, so you can at least navigate "near". These latter functionality is provided by an external call of synctex (see the predefined example configurations).

Forward search works both with DVI and PDF output. LyX simply checks which preview format you have used before (i. e., which format is already there in the temporary directory) and chooses the appropriate configuration for the respective format.

[5] For Sumatra CMCDDE is needed — downloadable from http://www.istri.fr/zip/CMCDDE.zip

6 Special Document Classes

6.1 A&A Paper

by Peter Sütterlin

6.1.1 Introduction

This section describes how LyX can be used to write articles for submission to the scientific journal *Astronomy and Astrophysics* (www.edpsciences.fr/aa/ http://www.edpsciences.fr/aa/) using Version 5.01 of the document class `aa.cls`. This package can be downloaded from the ftp site

ftp://ftp.edpsciences.org/pub/aa/readme.html

A manual comes together with that package, and this text is not meant to replace the original manual but is merely a short guide in how to realize the correct form of your paper.

Please note that the publisher of the journal was changed from Springer to EDP Sciences starting January 1, 2001. That change involved also some slight changes to the style files, namely the removal of the thesaurus command. The LyX class aa supports the newest version of these style files, V 5.01. If you have an older version installed, please upgrade. For compatibility, the old (version 4) layout has been kept as article (A&A V4). Please refer to the comments in `LyXDir/layouts/aapaper.layout`.

6.1.2 Getting started

It is recommended you start from the example template distributed with LyX. If you are not using a template, note the following settings:

- Select article (A&A) in the Document ▷ Settings dialog (OK, that one was obvious).
- Don't change the option Page style: Leave it set to default. The whole layout is done by the macros, you shouldn't change anything.

6.1.3 The header block

First thing to enter is the header information. It consists of seven entries, of which some are optional. They are

- Title: [required]
- Subtitle: [optional]
- Author: [required]
- Address: [required]
- Offprints: [optional] if more than one author: whom to contact for offprint requests.
- Mail: [optional] mail address for contacts.
- Date: [required]. Suggested format is `Received: <date>; Accepted <date>`

There is no need to issue the `\maketitle` command, this is done automatically by LyX when the header is finished. Although the order of the single header entries doesn't matter it is advised to keep the above sequence, just to get the best optics and meets the layout of the real document.

If you want to place footnotes in the header block, e. g. to state your present address, just use the standard footnote via the menu Insert ▷ Footnote. LyX will automagically use the term `\thanks{}` in that case.

In addition to these topics, the macros use three additional LaTeX commands that have no counterpart in LyX:

- `\and` to separate different names for more than one author and institute, respectively.
- `\inst{<nr>}`to mark corresponding author/institute pairs. The institutes are numbered sequentially as they appear in the Address field, so you have to put a marker to each author.
- `\email{address}` to supply an email address for fast contact.

In all cases, the appropriate command has to be entered in LyX and marked as LaTeX code. See the examples.

6.1.4 The abstract

The abstract should immediately follow the header block. With version 5 the abstract environment was changed to a command, and there is now a resctriction to only one paragraph. In addition, it should contain an entry with the keywords. This is not yet implemented for LyX, therefore you have to enter the LaTeX command `\keywords{}` by hand and mark it as LaTeX code. Refer to the example paper.

6.1.5 Supported environments

The A&A paper layout supports the following environments for structuring your text:

- Standard
- Section
- Subsection
- Subsubsection
- Itemize
- Enumerate
- Description
- Caption
- Abstract
- Acknowledgment
- Bibliography
- LaTeX

6.1.6 Commands not supported by LyX

Some commands are not yet supported by the paper (A&A) layout for LyX. Some have already been mentioned. For the sake of completeness, they are listed all together here:

- `\and`
- `\email`
- `\appendix`
- `\authorrunning`
- `\inst{}`
- `\keywords{}`
- `\object{}`
- `\titlerunning{}`

If you want to use any of these commands, you have to enter them yourself. **Do not forget to mark them as LaTeX code!**

6.1.7 Figure and Table Floats

LyX provides support for the necessary float environments figure, figure*, table and table*, therefore we won't tell much about it here. Refer to the *User's Guide*. Just remember that tables should be left-aligned. For that, select the table and change the alignment in Edit ▷ Paragraph Settings.

There is only one special thing: the figures with caption besides the figure. To create such a figure, you have to do the following:

1. Create a wide figure float: Insert ▷ Float ▷ Figure, then right click in the figure and select Span columns.
2. Enter your caption text.
3. Press Return to move the cursor above the caption.
4. Insert your figure
5. Position the cursor behind the figure and insert a horizontal fill: Insert ▷ Special Character ▷ Horizontal Fill.
6. Switch to LaTeX mode: M-c t.
7. Enter `\parbox[b]{55mm}{`. **Do not close the brace!**
8. Position the cursor behind the caption text, switch to LaTeX mode and insert the closing brace: M-c t }.

Also, refer to the figures in the example paper.

6.1.8 Referee layout

For submission, the paper has to be formated in a special double-spacing layout. For this purpose, you have to give the option `referee` to the documentclass. This must be done using the extra class options field in the Document ▷ Settings dialog. Just enter the string `referee` there.

6.1.9 The example paper

The Examples directory contains an example paper written with LyX. It is the example paper from the original macro package, translated to LyX. Use it for inspiration, and compare the original LaTeX code with LyX way of writing.

6.2 AASTeX

by Mike Ressler

6.2.1 Introduction

AASTEX is a set of macros produced by the American Astronomical Society to facilitate electronic manuscript submission to the three journals they publish: the Astrophysical Journal (including the Letters and Supplement), the Astronomical Journal, and the Publications of the Astronomical Society of the Pacific. LYX has proven to be an excellent tool for generating these documents, especially given its equation, citation, and figure handling capabilities. LYX requires version 5.0 (or higher) of these macros; preferably 5.2, which is the version described here, or higher. Versions prior to 5.0 are intended for use with LATEX2.09 and are fundamentally incompatible with LYX. The AASTEX package may be downloaded from the AASTEX Web site

http://www.journals.uchicago.edu/AAS/AASTeX

A complete user guide is contained in that package and you should familiarize yourself with it thoroughly before embarking on writing a paper in LYX. LYX will not reduce the need to figure out all the AASTEX commands, it will only reduce the drudgery of typing everything in. It is your responsibility to ensure that the final exported LATEX document conforms completely to the requirements of the journal to which you are submitting your paper.

6.2.2 Starting a New Paper

I strongly suggest that you start with the AASTEX template file. Click on File▷ New from Template, enter the new file name, then choose the `aastex.lyx` template. This will show the most common fields found in a manuscript. Simply overwrite the existing text (including the brackets, <>) with the correct information. Many of the AASTEX commands and environments can be implemented directly in LYX, but some cannot: most noticeably `\altaffilmark` and `\altaffiltext`, which should stick out like a sore thumb if you actually just opened the template file. For commands such as these, the LATEX code must be entered directly and marked as such. Such commands are referred to as TEX code, or Evil Red Text. I tried to minimize the amount of TEX code needed in an AASTEX document, but there is still a bit more required than any of us would like.

6.2.3 Finishing Your Paper

When the paper is finished to your satisfaction and previews/prints correctly, there are a few "postprocessing" actions which need to be done before you submit it to the journals.

1. Export your paper as a LATEX file (File▷ Export▷ LATEX).
2. Edit the resulting `.tex` file with your favorite text editor

a) remove the comment lines before the `\documentclass` command

b) remove the `\usepackage...{fontenc}` line if it appears (usually just after `\documentclass`}; also remove the `\secnumdepth` line if it appears.

c) remove everything between (and including) the `\makeatletter` and `\makeatother` commands, except for any commands you specifically put into the LaTeX preamble (which should appear immediately after the "User specified LaTeX commands" comment in the `.tex` file).

3. Run the resulting file through LaTeX to make sure it still processes correctly.

4. Reread the journal requirements to make sure your filenames and formats are correct.

5. Submit it.

6.2.4 Comments On Specific Commands

I will not describe the detailed usage of the individual AASTeX commands: the AASTeX User Guide (`aasguide.tex`) gives a good description of each. Thus it's probably easiest for me to go down the list as found in the guide and offer comments where necessary. So let's begin ...

6.2.4.1 Things that work as expected

Because they work as you might expect, I simply list them and the section they are found in: `\documentclass` (2.1.1), `\begin{document}` (2.2), `\title` (2.3), `\author` (2.3), `\affil` (2.3), `\abstract` (2.4), `\keywords` (2.5), `\section` (2.7), `\subsection` (2.7), `\subsubsection` (2.7), `\paragraph` (2.7), `\facility` (2.10), `\begin{displaymath}` (2.12), `\begin{equation}` (2.12), `\begin{eqnarray}` (2.12), `\begin{mathletters}` (2.12), `\begin{thebibliography}` (2.13.1), `\bibitem` (2.13.2), all the cite commands and their variations (2.13.2), the generic graphicx figure commands (2.14.1), `\begin{table}` (2.15.4), `\begin{tabular}` (2.15.4), `\caption` (2.15.4), `\label` (2.15.4, amongst other places), `\tablerefs` (2.15.5), `\tablecomments` (2.15.5), `\url` (2.17.4), `\end{document}` (2.18).

The following style options also work correctly: `longabstract` (2.4), `preprint` (3.2.1), `preprint2` (3.2.2), `eqsecnum` (3.3), `flushrt` (3.4). Simply put them in the Options box in Layout ▷ Document.

6.2.4.2 Things that work, but require more comment

The following items work, but require a little more discussion:

- These items are reserved for use by the journal editors, but you can put them into the LaTeX preamble if you feel compelled to do so: `\received`, `\revised`, `\accepted`, `\ccc`, `\cpright` (all from 2.1.3)

- These items may be placed in the LaTeX preamble, and are included as blanks in the template file: `\slugcomment` (2.1.4), `\shorttitle` (2.1.5), `\shortauthors` (2.1.5)
- `\email` (2.3) – can only be used "standalone", not in the middle of a paragraph. Use TeX code if you need to embed it.
- `\and` (2.3) – will have extra {} after it. This should not cause an error.
- `\notetoeditor` (2.6) – can only be used "standalone", not in the middle of a paragraph. Use TeX code if you need to embed it.
- `\placetable` (2.8) – can't insert a cross-reference tag, you must type the tag name by hand
- `\placefigure` (2.8) – same as for `\placetable`
- `\acknowledgements` (2.9) – will have extra {} after it. This should not cause an error.
- `\appendix` (2.11) – will have extra {} after it. This should not cause an error.
- `\figcaption` (2.14.2) – you can insert an optional filename argument by placing the cursor at the beginning of the text and selecting Insert ▷ Short Title. "Short Title" inserts an optional argument of the type needed by `\figcaption`. Hopefully it will be renamed someday.
- `\objectname` (2.17.1) – same as `\figcaption` for the catalog ID optional parameter
- `\dataset` (2.17.1) – same as `\figcaption` for the catalog ID optional parameter

6.2.4.3 Things not implemented, use TeX code

`\altaffilmark` (2.3), `\altaffiltext` (2.3), `\eqnum` (2.12), `\setcounter{equation}` (2.12), Journal name abbreviations (2.13.4), `\figurenum` (2.14.1), `\epsscale` (2.14.1), `\plotone` (2.14.1), `\plottwo` (2.14.1), `\tablenum` (2.15.4), `\tableline` (2.15.4, insert it as the first element in the lefthand cell after where you want it. Don't use any of LyX's rules in the table), `\tablenotemark` (2.15.5), `\tablenotetext` (2.15.5), much of Misc (2.17, except `\objectname`, `\dataset`, `\url`, and `\email`; see above), `\singlespace` (3.1), `\doublespace` (3.1), `\onecolumn` (3.2), `\twocolumn` (3.2)

6.2.4.4 Things that cannot be implemented

. . . at least in any meaningful sort of way, so I suggest ignoring them. They are the references environment (2.13.3), and the deluxetable environment (2.15). If you really, really need to use deluxetable, I suggest editing it in a separate file with a text editor, then using Insert ▷ Child Document to include it in your LyX document. See the `aas_sample.lyx` file to see an example of this.

6.2.5 FAQs, Tips, Tricks, and Other Ruminations

6.2.5.1 Getting LyX and AASTeX to cooperate

It can be a bit tricky to get LyX to recognize a new layout and document class. When all else fails, do this:

1. Make certain that LaTeX can find AASTeX. Copy sample.tex (and perhaps table.tex) from the AASTeX distribution into a directory completely unrelated to LaTeX or AASTeX and run LaTeX on `sample.tex`.
2. Make certain that `aastex.layout` appears in LyX's `layouts` folder
3. Rerun Tools ▷ Reconfigure in LyX, then restart LyX.
4. Open a regular new file, not from a template. Does AASTeX appear in the class list in Document ▷ Settings?

If you get a warning from an existing AASTeX document about not being able to find the AASTeX layout or a message about "You should not mix title layouts with normal ones", things haven't been installed correctly.

6.2.5.2 LaTeX error processing a table

LyX, by default, attempts to center the table caption/title. This seems to produce a bad interaction in AASTeX so you should click somewhere in the caption/title, then select Edit ▷ Paragraph Settings, then set the Alignment to Block. This took care of it for me.

6.2.5.3 References

A couple of things: 1) I have noticed some funny spacing in the reference entries in the text. When you enter the bibliography item data, make sure their is *no* space between the last author and the parenthesis setting off the year; *e. g.* type `Ressler(1992)`, not `Ressler (1992)`. 2) Entering the references at all is not obvious. The easiest thing is to start typing your first reference at the end of the document, then mark it as type References. That will put a small gray box in front of what you just typed. Click on the box to fill in the rest of the information. For new references, go to the end of an existing reference and press return. That will create a new line with its own box, etc.

6.2.5.4 Including EPS files

Even though AASTeX provides its own figure commands (`\plotone`, for example), I much prefer LaTeX's standard figure commands (with the default graphicx). You can insert the `\plotone`, etc. commands as TeX code into a Figure Float box if you desire, but I never have much luck getting the layout right. With the standard graphics,

LyX will insert a `\usepackage{graphicx}` command into the LaTeX preamble and handle the figures in the standard LaTeX 2_ε way, interspersing the figures in the text. I believe ApJ accepts figures exactly this way now; AJ might still use the "stack everything at the end" technique.

6.2.5.5 Things I could have done, but didn't

There are a few "pretty" things I could have implemented, but chose not to. For instance, I saw no point in double-spacing the text in the LyX window, even though it is double-spaced in the paper manuscript. Also, I chose not to make separate layouts for the preprint and preprint2 styles. Since I assume you will spend most of your time in the plain manuscript mode anyway, I decided not to chew up more disk space with this.

6.2.6 Final Caveat

Your mileage may vary. I've now had papers published by both ApJ and AJ that have had 98 % of the effort done in LyX; the last 2 % was the LaTeX post-processing and a few cleanups. I have had no trouble with the submission process, and I'm sure the journals were never aware that there might be a difference. So, go forth and publish!

6.3 AMS LaTeX

by David Johnson; updated by Richard Heck

The AMS LaTeX layouts are set up to conform to suggested styles for mathematical papers to be submitted to American Mathematical Society publications. The layouts are not tailored to a specific journal, but easily can be. You should refer to the AMS documentation for specific instructions for each journal (usually it will entail only changing a single line in the TeX output). That documentation is available on the Web at `http://www.ams.org` or by ftp at `ftp://ftp.ams.org/pub/tex/amslatex/`. These layouts are appropriate, and useful, for any mathematical writing.

There are two basic AMS LaTeX layouts:

- amsart: The standard AMS article format.
- amsbook: the standard AMS book (really, monograph) format.

The layouts themselves contain only the minimum necessary to use the AMS classes. They do not, in particular, contain any of the 'theorem' environments used for setting theorems, lemmas, and the like. These are contained, instead, in the `Theorems (AMS)` module, which is loaded by default when you select one of the AMS classes. (It

can also be used with other classes and can be removed, if you would rather use something else.) Less commonly used environments are in the `Theorems (AMS-Extended)` module, which must be loaded manually.

By default, theorems and the like are numbered consecutively throughout the document, but this may be modified by loading the module `Theorems (Order by Section)` or, if you are using `book (AMS)`, the module `Theorems (Order by Chapter)`. These will number the results as $n.m$, where the first number refers to the section (or chapter) and the second refers to the total number of results so far in that section (or chapter). Many environments are also available unnumbered. These are indicated by an asterisk at the end. If you happen to want *only* unnumbered results, the module `Theorems (Starred)` provides that option.

Note that these modules do not *have* to be used with the AMS classes. It is perfectly possible to use the `Theorems (AMS)` module, and the others mentioned, with other classes, such as `article`, `report`, `book (KOMA-script)`, and so forth.

6.3.1 What these layouts provide

There is a long list of included environments provided by these layouts. In AMS-LaTeX, there is, in fact, an opportunity to define an unlimited variety of 'theorem' environments. However, the AMS recommends the environments that are available in LyX.

The following environments—as well as the standard environments, such as SECTION, BIBLIOGRAPHY, TITLE, AUTHOR, and DATE—are provided by `article (AMS)` and `book (AMS)`:

Address This should be the author's permanent address.

Current Address This should be the author's temporary address at the time of submission, if different from the Address.

Email Author's e-mail address

URL Author's Web address, if desired.

Keywords Key words or phrases used to identify specific topics discussed in the paper.

Subjectclass These refer to the AMS Subject Classifications, published and described in *Mathematical Reviews*. These are also available online at the AMS cites listed above.

Thanks

Dedicatory

Translator

The following environments are provided by both the `Theorems` and `Theorems (AMS)` modules, in the latter case in both starred (unnumbered) and unstarred (numbered) versions. These same environments are provided only in the starred versions by the `Theorems (Starred)` module:

Theorem 1. *This is typically used for the statements of major results.*

Corollary. *This is used for statements which follow fairly directly from previous statements. Again, these can be major results.*

Lemma 2. *These are smaller results needed to prove other statements.*

Proposition 3. *These are less major results which (hopefully) add to the general theory being discussed.*

Conjecture 4. *These are statements provided without justification, which the author does not know how to prove, but which seem to be true (to the author, at least).*

Definition. Guess what this is for. The font is different for this environment than for the previous ones.

Example. Used for examples illustrating proven results.

Problem 5. It's not really known what this is for. You should figure it out.

Exercise. Write a description for this one.

Remark 6. This environment is also a type of theorem, usually a lesser sort of observation.

Claim. Often used in the course of giving a proof of a larger result.

Case 1. Generally, these are used to break up long arguments, using specific instances of some condition.

Case 2. The numbering scheme for cases is on its own, not together with other numbered statements.

Proof. At the end of this environment, a QED symbol (usually a square, but it can vary with different styles) is placed. If you want to have other environments within this one—for example, Case environments—and have the QED symbol appear only after them, then the other environments need to be nested within the proof environment. See the section *Nesting Environments* of the *User's Guide* for information on nesting. □

And these environments are provided by `Theorems (AMS-Extended)`:

Criterion. *A required condition.*

Algorithm. *A general procedure to be used.*

Axiom. *This is a property or statement taken as true within the system being discussed.*

Condition. Sometimes used to state a condition assumed within the present context of discussion.

Note. Similar to a Remark.

Notation. Used for the explanation of, yes, notation.

Summary 7. Do we really need to tell you?

Acknowledgement. Acknowledgement.

Conclusion. Sometimes used at the end of a long train of argument.

Fact 8. *Used in a way similar to Proposition, though perhaps lower on the scale.*

In addition, the AMS classes automatically provide the AMS LaTeX and AMS fonts packages. They need to be available on your system in order to use these environments.

6.4 AGU journals (aguplus)

by Martin Vermeer

6.4.1 Description

These are the layout files for some of the journals of the American Geophysical Society. It is assumed that you have both the AGU's own class files and AGUplus installed (everything to be found at`ftp://ftp.agu.org/journals/latex/journals`).

6.4.2 New styles

Redefined are Paragraph, Paragraph*. They are still called this in the LyX GUI, though their LaTeX equivalents in the AGU classes are Subsubsubsection and Subsubsubsection*.

Newly defined styles are Left_Header, Right_Header, Received, Revised, Accepted, CCC, PaperId, AuthorAddr, SlugComment. These are mostly manuscript attributes and defined in the AGU class documentation.

I suspect this is still badly incomplete.

6.4.3 New floats

Planotable and Plate. We also have a new Table_Caption.

6.4.4 Supported journals

- *Journal of Geophysical Research*: `jgrga.layout` — Martin Vermeer

Add your own, it isn't so hard! Look at the `jgrga.layout` example and `aguplus.inc`.

6.4.5 Bugs and things to remember

In order to use the new layouts, you must remember to do the following for a new document:

1. *Turn off babel.* This can be done in the Layout ▷ Document or Document ▷ Settings menu item. (AGU articles are always in English, right? So *don't* choose a language.)

2. Enter `jgrga` into the document's Extra Options field. (Yes, this is a bug.)

3. Make sure you use the `agu.bst` bibliography style, by entering `agu` into the second field of the BibTeX inset. None of the standard styles will do.

6.5 Broadway

by Garst Reese

6.5.1 Introduction

Broadway is for writing plays. The format is more decorative than Hollywood, and much less standardized. This format should be suitable for workshops.

6.5.2 Special problems

The same as in Hollywood.

6.5.3 Special features

Insert the Speaker names as labels then cross-reference the label to insert the name. The cross-reference dialog will show the current cast of characters.

6.5.4 Paper size and Margins

USLetter, left 1.6in, right 0.75in, top 0.5in, bottom 0.75in

6.5.5 Environments

The following environments are available. You can use broadway.bind to get the bind keys shown at the right.

- Standard
  You should not have to use this, but it is here for anything that does not fit otherwise.
- Narrative M-z n
  Used to describe stage setting and the action. First use of speaker names in all CAPs.
- ACT M-z a
  Automatically numbered. On screen it will be arabic, but will print as Roman.
- ACT* M-z S at
  Subtitle for ACT. It is just centered text.
- SCENE M-z S-S
  Not automatically numbered. You supply the number. This is because I couldn't figure out how.
- AT_RISE: M-z S-R
  A special case of Narrative to describe the setting and action as the curtain rises.
- Speaker M-z s
  The speaker's (actor's) title, centered in all CAPS.
- Parenthetical M-z p
  Instructions to the speaker. The parentheses are automatically inserted. The (will appear on screen, but both will be in the printed play. This environment is only used within Dialogue.
- Dialogue M-z d
  What the Speaker says.
- CURTAIN M-z S-C
  The curtain comes down.
- Title M-z S-T
- Author M-z S-A
- Right_Address M-z r

Hello there.

6.6 Dinbrief

The document class dinbrief can be used to type letters according to German conventions. A template file is included in `.../lyx/share/templates` for you to use as a starting point.

6.7 EGS journals (egs)

by Martin Vermeer

6.7.1 Description

This is the layout file for the European Geophysical Society journals. The needed `egs.cls` can be downloaded from the web site of the EGS under www.copernicus.org.

6.7.2 New styles

Right_address, Latex_Title, Affil, Journal, msnumber, FirstAuthor, Received, Accepted, Offsets. The current layout file is unfortunately very unmodular and would benefit from using the various `std*.inc` file inclusions.

6.8 Elsevier Journals

See the LyX template file `elsarticle.lyx`. It contains all infos you need.

6.9 Foils [aka FoilTEX]

by Allan Rae

6.9.1 Introduction

This section describes how to use LyX to make slides for overhead projectors. There are two document classes that can do this: the default slides class and the FoilTEX slides class. This section documents the latter.

I'm going to say this again, nice and clear, so that there's no misunderstanding:

This section documents the class "slides (FoilTEX)" *only.*

If you're looking for the documentation for "slides (default)", check out section 6.20. If your machine doesn't have the foils class ["slides (FoilTEX)"] installed, you'll probably have to use the default slides class, which isn't quite as good as foils.

The foils class is designed for use with version 2.1 of the foils.cls LaTeX class file which is now an integral part of LaTeX 2_ε.

6.9.2 Getting Started

Obviously, to use this document class, you need to select "slides (FoilTeX)" from the Class entry in the Document Layout dialog. There are some settings in the Document Layout dialog that you should know about that are specific to this class:

- Don't change the options Sides and Columns on the Document Layout dialog. They're ignored by the foils class.
- The default font size is 20 pt with the other options being 17 pt, 25 pt and 30 pt.
- The default font is sans serif but all math equations are still typeset in the usual roman font.
- FoilTeX supports A4 and Letter paper sizes as well as a special size for working with 35 mm slides. It doesn't support A5, B5, legal or executive paper sizes.
- Don't bother changing the Float Placement settings because they are ignored anyway. All floats appear where they are defined in the text.
- The Pagestyle setting behaves a bit differently for this class. FoilTeX provides extensive footer and header capabilities including a user-defined logo. See section 6.9.4.6 for more details. The title page is treated differently to all other pages in the document and is *always* unnumbered and *always* has the logo centered at the bottom of the page (if one is defined). The possible page style choices and what they do are as follows:

  empty The final output contains no page numbers, or other headers or footers (except footnotes of course).

  plain The final output contains page numbers centered at the bottom of the page. No other headings or footers (other than footnotes).

  foilheadings Page numbers in lower right corner. Additional headers and footers are also shown. This is also the default.

  fancy Gives you access to the fancyheadings package although its use with FoilTeX is discouraged by the writer of the FoilTeX package because of some potential page layout clashes.

6.9.2.1 Extra Options

The following options may be used in the extra class options in the Document ▷ Settings dialog.

35mmSlide This sets up the page layout for 7.33 in by 11 in paper, which is about the same aspect ratio as a 35 mm slide, making it a bit easier to work with this medium.

headrule Places a rule across the page below the header on every page except the title page.

footrule Places a rule across the page above the footer on every page except the title page.

dvips This is automatically set each time you create a new foils document. This option tells FoilTEX to use the dvips driver to rotate those pages that are set as landscape foils.

landscape Simply changes the page dimensions to those of a landscape page but doesn't do any rotation. Thus if you use this option you need to use an external program to rotate each page or feed your paper through your printer as landscape. Note that this option effectively reverses the roles of the Foilhead and Rotatefoilhead environments (don't worry these are described in the next section).

leqno Equation numbers on the left.

fleqn Flush-left equations.

6.9.3 Supported Environments

Most of the environments commonly supported in other classes are also supported by the foils class. There are several additional environments provided by FoilTEX as well as a couple added by LyX. The following environments are shared with other classes:

- Standard
- Itemize
- Enumerate
- Description
- List
- LyX-Code
- Verse
- Quote
- Quotation
- Title
- Author
- Date
- Abstract
- Bibliography
- Address
- RightAddress
- Caption
- Comment

That is, all the major environments apart from the sectioning environments. Since foils are essentially self-contained sections, with a title and body, FoilTEX provides specific commands for starting new foils and these are:

- Foilhead
- Rotatefoilhead

LyX also provides slightly modified versions of these two environments called:

- ShortFoilhead
- ShortRotatefoilhead

and the differences will be explained in the next section.

Since foils are often used in presenting ideas or new theorems and such FoilTEX also provides a comprehensive box of goodies for presenting them:

- Theorem
- Lemma
- Corollary
- Proposition
- Definition
- Proof
- Theorem*
- Lemma*
- Corollary*
- Proposition*
- Definition*

The starred versions are unnumbered while the unstarred versions are numbered. There are also two list environments added by LyX and these are:

- TickList
- CrossList

FoilTEX provides some powerful header and footer capabilities that are best set in the preamble although they may be set at any point in a document. If you want to change these settings in your document the best place to do so is at the very top of a foil, i. g. straight after the foilhead.

For this purpose, the following command styles are provided [Martin Vermeer]:

- My Logo
- Restriction
- Right Footer
- Right Header
- Left Header

There are also a few commands provided by FoilTEX that aren't directly supported by LyX but I'll tell you what they do and how to use them in section 6.9.5.

6.9.4 Building a Set of Foils

This section will give a simple introduction to using the different environments to build a set of foils. If you want to see an example set of foils, take a look at the `Foils.lyx` file you find in LyX's `examples` folder.

6.9.4.1 Give It a Title Page

Unlike other classes that provide Title, Author, Date and Abstract environments, foils creates the title on a page of its own. If you leave out the Date environment LaTeX will substitute the current date (every time you regenerate the output).

6.9.4.2 Start a New Foil

As I mentioned earlier, there are four ways of starting a new foil. For portrait foils you should use Foilhead or ShortFoilhead. The difference between these two environments is the amount of space between the title of the foil (the foilhead) and the body of the foil.

Landscape foils are generated using the Rotatefoilhead and ShortRotatefoilhead environments. Again the only difference is the spacing between foilhead and body. Both of the short versions have 0.5 inches less separation between the foilhead and the body.

One problem with the support for landscape foils is the requirement that you have to use the `dvips` driver to generate the PostScript output otherwise the foils won't be rotated. It is possible to get landscape foils even if you haven't got the `dvips` driver provided you can feed your foils sideways through your printer ;-)

6.9.4.3 Theorems, Lemmas, Proofs and more

Due to a small bug in LyX you can't have two of the same type of these environments directly following each other. They must be separated by something. If you try, you will just be extending the previous environment as if you had merged the two environments together. So, how do you get around this problem? The simplest option is to insert some text between the two environments or add a LaTeX environment between the two with just a "%" in it. This will force LyX to produce two separate environments and hence the correct LaTeX output. An example is provided in the example file included with the LyX distribution. Remember, this problem only occurs if you are trying to place two of the same type of theorem-like environments one directly after the other.

6.9.4.4 Lists

You get all the commonly supported list styles found in other classes as well as two new ones. I'll only describe the new ones here. If you want to find out more about the other list environments check out the *User's Guide.* If you intend to use itemized

lists you might also want to read about the Itemize Bullet Selection dialog described above in section 3.5.

The two new list styles, TickList and CrossList, are designed to make it easier for you to create lists of do's and don'ts or right and wrong by providing dedicated environments that use a tick or a cross as the label of the list. These lists are in fact dedicated variants of the Itemize environment. They do however require that you have the `psnfss` packages installed.

6.9.4.5 Figures and Tables

FoilTEX redefines the floating tables and figures so that they appear exactly where they are in the text rather than pushing them to the top of the page or to some user specified location. In fact if you change the float placement settings they are simply ignored.

6.9.4.6 Page Headers and Footers

My Logo and Restriction are two commands used to control the left-footer text string. The first is meant to allow you to include a graphic logo on your foils and defaults to "-Typeset by FoilTEX-". While the second is meant to provide a classification for the audience, *e. g.* Confidential. It is empty by default.

The remaining page corners can be filled by Right Footer (which defaults to page numbers), Right Header (top right) and Left Header (top left).

6.9.5 Unsupported FoilTEX Goodies

All the commands mentioned below need to be set in a LATEX environment or as TEX within another environment.

6.9.5.1 Lengths

All lengths are adjusted using the `\setlength{`*lengthname*`}{`*newlength*`}` command. Where *lengthname* should be replaced by the name given to the length you want to change and *newlength* is the length value. All lengths should be specified in units of length such as inches (`in`), millimeters (`mm`) or points (`pt`) or relative to some document or font-based length such as `\textwidth`.

It's possible to change the spacing between a foilhead and the body of the foil by adjusting the length specified by `\foilheadskip`. For example, to make *all* foilheads 0.5 in closer to their bodies put the following in the preamble: `\setlength{\foilheadskip}{-0.5in}`

The spacings around floats can be adjusted by setting these lengths:

`\abovefloatskip`	Separation between the text and the top of the float
`\abovecaptionskip`	Separation between the float and the caption
`\belowcaptionskip`	Separation between the caption and the following text

`\captionwidth`	You can make the captions narrower than the surrounding text by adjusting this length. Best done relative to `\textwidth`.

There are also several title page related lengths that you may find useful if you have a long title or several authors:

`\abovetitleskip`	Separation from headers to Title
`\titleauthorskip`	between Title and Author environments
`\authorauthorskip`	between multiple Author lines
`\authordateskip`	between the Author and the Date
`\dateabstractskip`	between the Date and the Abstract

The last length related command affects all the list environments. If you place `\zerolistvertdimens` *inside* a list environment then all the vertical spacing between the list items is removed. Note that this is a command not a length so it doesn't require `\setlength` like the stuff mentioned above.

6.9.5.2 Headers and Footers

The `\LogoOn` and `\LogoOff` commands control whether the logo in the MyLogo definition appear on a given page. If you put `\LogoOff` in the preamble then none of the foils will have the logo on them. If you don't want the logo on a particular page place the `\LogoOff` directly after the foilhead of that page and the `\LogoOn` directly after the next foilhead.

If you decide to use the fancy page style setting in the Document Layout dialog you should probably add `\let\headwidth\textwidth` to your preamble so headers and footers on landscape pages are correctly placed when rotated. This is due to some clashes between the page layouts provided by the fancyheadings package and the foils class.

6.10 Hollywood (Hollywood spec scripts)

by Garst Reese

6.10.1 Introduction

Getting the format of a Hollywood script right is a "rite of passage." It is designed to make the readers focus on content and to be easy and familiar for the actors to read. Each page of a script should be one minute of film. Nothing goes in a script that you cannot see or hear on screen. The courier 12 pt font should be used throughout. No italics.

6.10.2 Special problems

Speakers' lines should NEVER break in mid-sentence. If a speaker's lines continue over a page break, repeat the Speaker title followed by (Cont'd).

6.10.3 Special features

Insert the Speaker names as labels then cross-reference the label to insert the name. The cross-reference dialog will show the current cast of characters. You can use this to insert the speaker name in narratives also.

6.10.4 Paper size and Margins

USLetter, left 1.6in, right 0.75in, top 0.5in, bottom 0.75in

6.10.5 Environments

The following environments are available. You can use hollywood.bind to get the bind keys shown at the right.

- Standard
  Used where nothing else works. Try to avoid it.
- FADE_IN: M-z S-I
  Usually followed by something like "on Sally waking up."
- INT: M-z i
  Introduces a new INTERIOR camera set-up. Always followed by DAY or NIGHT, or something similar to define the lighting required. Everthing on this line in CAPS.
- EXT: M-z e
  Introduces a new EXTERIOR camera set-up. Everthing on this line in CAPS.
- Speaker M-z s
  The character speaking.
- Parenthetical M-z p
  Instructions to the speaker. The () are automatically inserted, but only the (will show in LyX. Both will be printed.
- Dialogue M-z d
  What the Speaker says.
- Transition M-z t
  Camera movement instruction. e. g. CUT TO:

- FADE OUT: M-z S-I
- Author M-z S-A
- Title M-z S-T
- Right_Address M-z r

6.10.6 Script jargon

- (O.S) — off screen
- (V.0) — voice over
- b. g. — background
- C.U. — close-up
- PAN — camera movement
- INSERT — cut to close-up of

6.11 ijmpc and ijmpd

by Panayotis Papasotiriou

6.11.1 Overview

The ijmpc package is a set of macros that facilitates electronic manuscript submission to the *International Journal of Modern Physics C*. Similarly, the ijmpd package is for creating manuscripts to be submitted to the *International Journal of Modern Physics D*. Both journals are published by World Scientific. The corresponding document classes are named `ws-ijmpc.cls` and `ws-ijmpd.cls`, respectively. These files, together with instructions for the authors, can be downloaded from the sites http://www.worldscinet.com/ijmpc/mkt/guidelines.shtml and http://www.worldscinet.com/ijmpd/mkt/guidelines.shtml. Both packages are modified versions of the standard "article" package, and they are almost (but not exactly) identical. Most of their features are supported by LyX. I have used LyX successfully to write articles submitted to both journals without any problem.

6.11.2 Writing a paper

As usual, the easiest way to write a paper is to start with a template. Click on File▷ New from Template, then choose the `ijmpc.lyx` or `ijmpd.lyx` template. This will give an (almost) empty document that includes the most common fields found in a manuscript. Simply overwrite the existing text (including the brackets, <>) with your text. You should keep in mind the following remarks.

1. LyX won't let you change the font size and the page style of the document, because such modifications are not allowed by both packages.

2. The language of the document should not be changed. Before previewing your paper, be sure that the babel package is not used. To do this, click on Tools ▷ Preferences, select the Lang Opts tab, deselect the Use babel checkbox in the language settings, and click on Apply (or Save, if you wish to make this change permanent).

3. The "Keywords" style must be used to define keywords.

4. The ijmpc package provides a style named "Classification Codes", which can be used to define classification codes, such as PACS numbers. Note that this facility is not supported by the ijmpd package.

5. Several new environments are available: "Definition", "Step", "Example", "Remark", "Notation", "Theorem", "Proof", "Corollary", "Lemma", "Proposition", "Prop", "Question", "Claim", and "Conjecture". Their use is more or less obvious. LyX supports all these environments; it will use the proper label, text style, and numbering scheme for each of them.

6. Both packages use basic citations; the natbib package should not be used. In LyX, citation references are shown as usual; in the output, citations are shown as superscripts. If you want to use a citation as normal text, you should use the `refcite` command, e. g. "See Ref. `\refcite{key}`".

7. There is no "Acknowledgments" section in both packages. To put acknowledgments, just use the "Section*" environment.

8. Appendices may be added to the paper, *after* the Acknowledgments and *before* the References. LyX provides a special environment, called "Appendices Section" which marks the beginning of the appendices. This environment should be left blank; it just sends a LaTeX command, but nothing is really printed. In LyX, the word "Appendix" is printed with blue letters, as a signal that all sections after that point are appendices. To write an appendix, use the "Appendix" environment. LyX will number each appendix with capital letters, as required by both journals. Note that "Appendices Section" *must* be present before the first appendix; if not, all appendices will be numbered as normal sections in the output.

9. The ijmpc and the ijmpd packages use the `tbl` command to implement table captions. As a result, a table created by LyX is printed correctly, but its caption is ignored. However, you can use some TeX code to overcome this problem, so that captions are printed as expected. To do so, create a float table as usual, remove the caption, and replace it with the TeX code `\tbl{`*`your table caption`*`}{` (sic); you must also the TeX code `}` immediately after the tabular

material. Study the example table included in the template files to see how this trick is implemented. Alternatively, If you need table captions, you should implement the whole table float in a `.tex` file, then include this file to the LyX document (Insert ▷ File ▷ Child Document). Details on how to create a table float can be found in the files `ws-ijmpc.tex` and `ws-ijmpd.tex`, included in the corresponding packages.

6.11.3 Preparing a paper for submission

Before you submit your paper you must export the LyX document as a LaTeX file (File ▷ Export ▷ LaTeX)[1], then make the following changes to the resulting `.tex` file.

1. Remove the comment lines before the `\documentclass` command.
2. Remove everything between (and including) the `\makeatletter` and `\makeatother` commands, except for any commands you specifically put into the LaTeX preamble.

The modified `.tex` file should be saved and processed through LaTeX as many times as necessary. You may also want to check the resulting `.dvi` document.

6.11.4 Use of TeX code

The use of TeX code is reduced to two commands, which must be placed at the top of the document. If you started writing your paper by using the `ijmpc.lyx` or the `ijmpd.lyx` template, the TeX code needed is already in its place; you usually don't need to delete it. You may only modify the first TeX code to specify the information printed to the top of odd and even pages (authors' names and short paper's title, respectively). This TeX code must have the form `\markboth{Authors' Names}{Short Paper's Title}`.

6.12 iopart

by Uwe Stöhr

6.12.1 Overview

The iopart package provides a document class to create electronic manuscript submission to the journals published by the Institute of Physics. Instructions for the authors how to create a paper using the iopart class can be downloaded together with the iopart package from the site ftp://ftp.iop.org/pub/journals/latex2e.

[1] Actually you have the choice between LaTeX (plain) and pdflatex. If you intend to use pdflatex to prepare the paper, you should use the pdflatex option so that included graphics are converted to PDF format, ready for use by pdflatex.

6.12.2 Writing a paper

The easiest way to write a paper is to start with the file *IOP-article.lyx* that is available in LyX's examples files folder. Open this file, save it under a new name, and start writing. The example file explains how to use the special text environments. Here are the most important advices:

- To be able to compile your document to a PDF, PS, or DVI, ensure that the two options Use AMS math package in the document settings under Math Options are not used!
- The title environment defines the kind of your paper. So use one of the following environments for the title:
  - Title for a Paper
  - Review for a Review
  - Topical for a Topical review
  - Comment for a Comment
  - Note for a Note
  - Paper for a Paper (same as Title)
  - Prelim for a Preliminary communication
  - Rapid for a Rapid communication
  - Letter for a Letter to the editor
- All title environments except of Letter can have an optional short title.
- There is a general title environment Article which is not directly supported by the LyX. This can be used as TeX code when your document doesn't fit into one of the other title types.

For more informations like hints for special table and formula formatting, look at the IOP author guidelines.

6.13 Kluwer

by Panayotis Papasotiriou

6.13.1 Overview

The Kluwer package is a set of macros produced by Kluwer Academic Publishers that facilitates electronic manuscript submission to the journals they publish. Most known of them (at least in my domain of interest) are *Astrophysics and Space Science* and *Solar Physics*, but there are many others (see a complete list at http://www.

wkap.nl/jrnllist.htm/JRNLHOME). The Kluwer package may be downloaded from the site http://www.wkap.nl/kaphtml.htm/STYLEFILES. A complete user guide is contained in that package (but it can also be downloaded separately).

LyX supports many features of the package but not everything. However, the TeX code needed is reduced to some "peculiar" commands of the package (see 6.13.4). I have recently used LyX to write an article submitted to the *Astrophysics and Space Science* without any problem.

6.13.2 Writing a paper

The easiest way to write a paper is to start with the Kluwer template file. Click on File ▷ New from Template, then choose the `kluwer.lyx` template. This will give an (almost) empty document that includes the most common fields found in a manuscript and a short description of their use. As in most templates, simply overwrite the existing text (including the brackets, <>) with the correct information.

6.13.3 Preparing a paper for submission

As in the AASTeX package, before you submit your paper to a journal you must "postprocess" it as follows.

1. Export your paper as a LaTeX file. To do this, click on File ▷ Export ▷ LaTeX.
2. Edit the resulting `.tex` file with a text editor and make the following changes
   a) remove the comment lines before the `\documentclass` command,
   b) remove everything between (and including) the `\makeatletter` and `\makeatother` commands, except for any commands you specifically put into the LaTeX preamble.

   Save the resulting `.tex` file.
3. Run the `.tex` file through LaTeX as many times as necessary (usually up to three).
4. View the resulting `.dvi` document using, e. g. xdvi, and check if everything is OK (it should, if you didn't make any mistake).

6.13.4 "Peculiarities" of the Kluwer package

The Kluwer package has the following "peculiarities".

1. It is possible to write multiple articles in the same LaTeX file[2]. Each article must be included in the environment "article". Unfortunately, this environment cannot be omitted, even if you write just one article. Therefore, each article starts

[2] I can't imagine any good reason to do this.

with the command `\begin{article}` and, obviously, ends with the command `\end{article}`. Although this can be implemented in LyX, I didn't included it, since it looks ugly and can confuse the novice user. Therefore, you need to enter them directly and mark them as LaTeX code (the well-known "TeX code").

2. Information given at the beginning of the article (i. g. title, subtitle, author, institution, running title, running author, abstract and keywords) must be included in an environment called "opening". This is not implemented in LyX, so you must enter title, subtitle etc. between two TeX code lines (`\begin{opening}` and `\end{opening}`).

3. According to the user manual, the label of each bibliography item must be written as `\protect\citeauthoryear{`*author(s)*`}{`*year*`}`.

The `kluwer.lyx` template takes care of all these "peculiarities". If you start a new paper using this template you don't need to do anything special. Just

1. don't delete the TeX code included in the template, and

2. copy the example bibliography item included in the template and modify it as necessary to enter new bibliography items.

6.14 Koma-Script

by Bernd Rellermeyer

6.14.1 Overview

The LyX document classes *article (koma-script)*, *report (koma-script)*, *book (koma-script)*, and *letter (koma-script)* correspond to the LaTeX document classes `scrartcl.cls`, `scrreprt.cls`, `scrbook.cls`, and `scrlettr.cls`, resp. of the Koma-Script family. They are replacements for the standard document classes `article.cls`, `report.cls`, `book.cls` and `letter.cls`, resp., and fit better to European typography conventions in a number of points.

- Standard character size is 11pt in *article (koma-script)*, *report (koma-script)*, and *book (koma-script)*, and 12pt in *letter (koma-script)*.

- Headings, labels of the description environment, and a number of elements of the *letter (koma-script)* document class are set in a bold sans serif font.[3] The

[3] There is a big difference between the bold sans serif old cm fonts and new ec fonts, especially in the appearance of headings. In comparison, the ec bold sans serif fonts look a bit thin. Here the LaTeX package `cmsd.sty` by Walter Schmidt helps to produce the "usual" appearance when using the ec fonts.

numbering of chapter headings is made in the same way as the numbering of section headings, that is without the extra line "Chapter...". In addition, the appearance of the headings can be modified by using a number of options (in LyX to be entered in the field Extra Options of the dialog Layout ▷ Document).

- The main means in the Koma-Script document classes to design the type area are the options BCOR and DIV (in LyX to be entered in the extra class options field in the dialog Document ▷ Settings). They make a clearer modification of page margins possible as do the options of the dialog Document ▷ Settings.
- The LaTeX document classes of the Koma-Script family define a number of additional commands. Those part of it which makes sense in LyX is implemented in corresponding paragraph types.

Detailed descriptions of the LaTeX document classes of the Koma-Script family can be found in the Koma-Script documentation *scrguide* (German) and *scrguien* (English).

6.14.2 article (koma-script), report (koma-script), and book (koma-script)

The document classes *article (koma-script)*, *report (koma-script)*, and *book (koma-script)* are implemented in the layout files `scrartcl.layout`, `scrreprt.layout`, and `scrbook.layout`, resp. They contain all the paragraph types of the corresponding standard document classes *article*, *report*, and *book*, resp., partly modified, with the exception of the LyX specific List-type, which is replaced by the new Labeling-type having the same functionality. Beside the Labeling-Type there is a number of new paragraph types added. They are *not* part of *letter (koma-script)*.

- Addpart, Addchap, Addsec: are equivalents to Part*, Chapter* and Section*, resp., additionally inserting an entry in the table of contents. Addpart and Addchap are not contained in *article (koma-script)*.
- Addchap*, Addsec*: behave exactly as Addchap and Addsec, resp., additionally clearing running heads. Addchap* is not contained in *article (koma-script)*.[4]
- Minisec: generates a heading directly above the following paragraph in the standard character size without affecting the structure of the document.
- Captionabove and Captionbelow are special captions which respect the different space settings needed for captions placed above or below an element (if you follow strict typographic rules, you might want to place table captions always above the table). You can also use the class option `tablecaptionsabove`, which will switch caption to captionabove for tables and captionbelow for figures. You need at least Koma-Script version 2.8q to use this.

[4] There is also an `\addpart*` command in *book (koma-script)* and in *report (koma-script)*, but since this is identical to Part*, is has not been implemented in LyX.

- Dictum: can be used to set a bonmot, e. g. at the beginning of a chapter. If you use the optional argument (Insert ▷ Short Title), you can insert the dictum's author there. Dictum and author are separated by a line. You need at least Koma-Script version 2.8q to use this. Dictum is not contained in *article (koma-script)*.

The following types, together with the standard types Title, Author, and Date, form the title area of the document. They must be entered ahead of the first "ordinary" paragraph.[5] When such a type is used more than once, the latter usage overwrites the former one, that means, for every type only the latest usage is valid. The order of the different types however has, like Title, Author, and Date, no effect on the appearance of the produced document.

- Subject: produces a centered paragraph above the ordinary title (Title, Author, Date) for the subject of the document.

- Publishers: produces a centered paragraph below the ordinary title (Title, Author, Date) for the publishers' name.

- Dedication: in *report (koma-script)* and *book (koma-script)* produces a centered paragraph on its own page behind the title page, or in *article (koma-script)* produces a centered paragraph below the ordinary title (Title, Author, Date, Publishers) for a dedication.

- Titlehead: produces a left aligned paragraph above the ordinary title (Title, Author, Date, Subject) for a document‘s head.

- Uppertitleback: produces in a double-sided print in *report (koma-script)* and *book (koma-script)* a left-aligned paragraph at the top of the title page‘s back or has no effect in a single-sided print or in *article (koma-script)*.

- Lowertitleback: produces in a double-sided print in *report (koma-script)* and *book (koma-script)* a left-aligned paragraph at the bottom of the title page‘s back or has no effect in a single-sided print or in *article (koma-script)*.

- Extratitle: produces a special "dirty" page ahead of the actual document containing a paragraph without special formatting.

The layout files for the document classes *article (koma-script)*, *report (koma-script)*, and *book (koma-script)* do include the file `scrmacros.inc`. This is thought of as a place to define your own types. Copy scrmacros`.inc` in your personal layout directory and edit the file!

[5] The corresponding LaTeX commands must appear before the `\maketitle` command.

6.14.3 letter (koma-script)

The document class *letter (koma-script)* is implemented in the layout file `scrlettr.layout`. It contains all the paragraph types of the corresponding standard document class *letter*, partly modified, with the exception of the LyX specific types LyX-Code and Comment and the List type, which is replaced by the new Labeling type. In addition, it contains, in contrast to the standard document class, the standard types LaTeX, Quotation, Quote, and Verse. Furthermore, there are a number of new letter specific types.

The appearance of the letter produced by this document class can be controlled by a number of LaTeX commands, which you can put in the LaTeX preamble.[6] A detailed German description of such LaTeX commands can be found in the Koma-Script documentation *scrguide*. With it, the letter's author can produce his personal letter layout.

The types Letter and Opening define the beginning of the letter and must be used in every letter. To emphasize them in the LyX document class, they are marked with the letter *L* or *O*, resp. in the left margin. It is possible to write any number of letters in one file. An Opening type produces a new letter using the same addressee and a Letter type produces a new addressee. The types Closing, PS, CC, and Encl are ordinary paragraph types and can also be used several times in one and the same letter.

- Letter: produces a paragraph for the addressee and implicitly defines the beginning of the letter.
- Opening: produces a paragraph for the form of address and implicitly produces a new letter.
- Closing: produces a paragraph for a close.
- PS: produces a paragraph for a postscript.
- CC: produces a paragraph for a distribution list.
- Encl: produces a paragraph for enclosures.

The types Name, Signature, Address, Telephone, Place, Backaddress, Specialmail, Location, Title, and Subject are input types provided with a label to enter information,

[6] For example, the standard appearance of the letter's heading, consisting of name and address, is quite self-willed. An "ordinary" heading is produced by the following LaTeX commands in the preamble:

```
\firsthead{\parbox[b]{\textwidth}
  {\ignorespaces \fromname\\ \ignorespaces \fromaddress}}
\nexthead{\parbox[b]{\textwidth}
  {\ignorespaces \fromname \hfill \ignorespaces \pagename\ \thepage}}
```

which will be processed by the document class.[7] The types must be used ahead of the corresponding Opening type.

An implementation of these types in a WYSIWYG fashion does not seem to make sense, because the real appearance of the produced letter does not only depend on the usage of the particular type, but also on other factors. For example, a signature entered in the Signature type will in the standard behavior appear in the produced letter only, when in the same letter also a Closing type is used. The entered value of the Telephone type will in the standard behavior not appear in the produced letter at all. The possibility to design the letter‘s heading freely is already indicated in a footnote above.

The input types can also be used as empty paragraphs. This makes sense e. g. for the Signature type. If the Signature type is not used at all, in the standard behavior the value of the Name type is used as signature, whereas if an empty Signature type is used, no signature value is defined.

By using the input types it is possible to write a letter template, containing filled input types with your personal dates (name, address, etc.) and empty input types for other dates you want to enter.

- Name: sender’s name, in the standard behavior appears as a centered paragraph in small caps in the letter‘s heading.
- Signature: sender’s signature, in the standard behavior appears below the Closing type. If no Signature type is used, the value of the Name type appears instead.
- Address: sender’s address, in the standard behavior appears in a centered paragraph in the letter‘s heading below the sender’s name.
- Telephone: sender’s telephone number, in the standard behavior only sets the LaTeX variable `\telephonenum`.
- Place: place of the letter‘s making.
- Date: date of the letter‘s making. Place and Date, in the standard behavior, produce the place and the date in a right-aligned line below the addressee’s field. If an empty Date type is used, neither place nor date appear, independent of the value of the Place type. If no Date type is used, the date of the letter ‘s production is used.
- Backaddress: sender‘s back address, in the standard behavior appears above the addressee’s field in a small sans serif font.

[7] It could be seen as a matter of inconsequence, that the types Letter and Opening described above are not such input types as well. Because of the special meaning of those types, however, I have implemented them as ordinary paragraph types with a one letter mark in the left margin. Moreover, it would affect my feeling of symmetry, if the Opening type and the Closing type had such a serious different appearance.

- Specialmail: special mail information, in the standard behavior appears underlined above the addressee's field below the back address.
- Location: additional information, in the standard behavior appears on right side below the addressee‘s field.
- Title: the letter's title, in the standard behavior appears in a big, bold, sans serif font above the subject.
- Subject: the letter's subject, in the standard behavior appears in a bold font above the Opening paragraph.

The types Yourref, Yourmail, Myref, Customer, and Invoice produce a business letter like line above the Title line containing the fields "Your ref.", "Your letter of", "Our ref.", "Customer no.", "Invoice no.", and "Date". For the date field, the value of the Date type is used. If one of these "business letter types" is used, the value of the Place type however does not appear, but only the LaTeX variable `\fromplace` is set. The ordinary output of place and date in a right-aligned line below the addressee‘s field is suppressed. The types are implemented as input types provided with a label and must be used ahead of the corresponding Opening type.

- Yourref: Your ref.
- Yourmail: Your letter of.
- Myref: Our ref.
- Customer: Customer no.
- Invoice: Invoice no.

6.14.4 The new letter class: letter (koma-script v.2)

by Jürgen Spitzmüller

Koma-Script version 2.8 has introduced a new letter class `scrlttr2` which supersedes the now unsupported `scrlettr`. It has — on the LaTeX side — a completely new interface and is not compatible with the old class. Therefore, LyX supports both, though it is recommended you use the new class.

This class covers the same functionality as *letter (koma-script),* and a few more. The basic items are Address (receiver's address, same as Letter in the old layout), Opening, and Closing. NextAddress will start a new letter (i. g. you can write several letters per document). New elements are sender's E-Mail, URL, Fax, Bank and the possibility to use a Logo (via Insert ▷ Graphics) in the header.

The biggest improvement is, though, that the letter's layout is configurable to meet almost any needs. This can be done via the preamble or with a special style

file (Letter Class Option, extension `*.lco`), that will be read in as a class option.[8] Have a look at the *koma-letter2* template that is included in LyX for examples. A detailed description is to be found in the Koma-Script documentation (*scrguide*).

6.14.5 Problems

Visualizing the Koma-Script document classes in LyX, the LyX internals cause some problems.

- The chapter number of a Chapter type appears on a line of its own above the chapter heading instead of appearing in the same line ahead of it. The cause for that is the LyX internal behavior for the labeltype Counter_Chapter in the layout file.
- The headings of the types Addchap and Addsec are only put in the "true" LaTeX table of contents, but not in the LyX table of contents (Document ▷ Table of Contents).
- The paragraphs in a *letter* document class appear in a skip separation mode, not indented. This is the standard behavior, no special LaTeX commands are needed for that. But in the Document ▷ Settings dialog the corresponding radio button indicates Indent. A Skip value always has the effect that extra LaTeX commands are inserted in the document to produce the gap, which is not what is wanted in this case.

6.15 Memoir

By Jürgen Spitzmüller

6.15.1 Overview

Memoir is a very powerful and constantly evolving class. It has been designed with regard to fictional and non-fictional literature. Its aim is to let the user have maximum control over the typesetting of his document. Memoir is based on the standard book class, but it can also emulate the article class (see below).

Peter Wilson, the developer of Memoir, is known as the author of lots of useful packages in the LaTeX world. Most of them have been merged with Memoir. Therefore, it is much easier to layout the table of contents, appendices, chapter designs and such. LyX, though, does not support all of these goodies natively. Some of them might be added to forthcoming releases[9], lots will probably never, due to the

[8] The KOMA package comes with some default `*.lco` files. There is, for instance, a `DIN.lco` file that follows german typesetting rules, or a `KOMAold.lco` that provides the default layout of the old `scrlettr` class. The latter can be loaded with the class option `KOMAold`, inserted via the Layout ▷ Document ▷ Extra Options field.

[9] You are invited to send suggestions to lyx-devel@lists.lyx.org.

limitations of LyX's framework. Of course you can still use all features with the help of some native LaTeX commands (TeX code[10]). In this section, we can only list those features which are natively supported by LyX. For detailed descriptions (and for the rest of features) we recommend you have a look at the detailed manual of the Memoir class[11], which is not only a user guide for the class, but also both a comprehensive description on good typesetting and a superb example for good typesetting itself.

6.15.2 Basic features and restrictions

Memoir supports basically all features of the standard book classes. There are, however, some differences, as follows:

Font sizes: Memoir has a broader range of font sizes: 9, 10, 11, 12, 14, 17

Page style: The fancy page style is not supported, due to a command clash between Memoir and the fancyhdr package (they both define a command with the same name, which confuses LaTeX). Instead, Memoir comes with a number of its own page styles (see Layout ▷ Document ▷ Page Style). If you want to use these for the chapter pages, you have to use the command `\chapterstyle` in the main text or in preamble (e. g. `\chapterstyle{companion}`).

Sectioning: Sectionings (chapter, section, subsection etc.) come with an optional argument in the standard classes. With this, you can specify an alternative version of the title for the table of contents and the headers (for instance, if the title is too long). In LyX, you can do this via Insert ▷ Short Title at the beginning of a chapter/section. Memoir features a second optional argument and thus separates the table of contents from the header. You can define three variants of a title with this: one for the main text, one for the table of contents, and one for the headers. Simply insert two optional arguments if you need this feature, the first one containing the short title for the Table of Contents, the second one containing an alternative short title for the headers.

TOC/LOT/LOF: In the standard classes (and in many other classes), the table of contents, the list of figures and the list of table start a new page automatically. Memoir does not follow this route. You have to insert a page break yourself, if you want to have one.

Titlepage: For some unknown reason, Memoir uses pagination on the title page (in the standard classes, title pages are "empty", i. g. without pagina). If you want an empty title page, type `\aliaspagestyle{title}{empty}` in the preamble.

Article: With the class option *article* (to be inserted in Layout ▷ Document ▷ Extra Options), you can emulate article style. That is, counters (footnotes, figures, tables

[10] Cf. section 2.3 for details.
[11] Cf. CTAN:/macros/latex/memoir/memman.pdf.

etc.) will not be reset on new chapters, chapters don't start a new page (but are—in contrary to "real" article classes—still allowed), parts, though, use their own page, as in book.

Oldfontcommands: By default, Memoir does not allow the use of the deprecated font commands, which have been used in the old LaTeX version 2.09 (e. g. `\rm`, `\it`). It produces an error and stops LaTeX whenever such a command appears. The class option *oldfontcommands* reallows the commands and spits out warnings instead (which does at least not stop LaTeX). Since a lot of packages and particularly BibTeX style files are still using those commands, we have decided to use this option by default.

6.15.3 Extra features

We will only describe the features supported by LyX (which is not much currently). Please consult the Memoir manual[12] for details.

Abstract: You may wonder why an abstract is an extra feature. Well, it is in book class. Usually books don't have abstracts. Memoir, however, has. You can use it wherever and how often you like.

Chapterprecis: You may know this older typesetting style: The contents of a chapter are summarized below the title and also in the table of contents (e. g. *Our hero arrives in Troia; he loses some friends; he finds others*). Chapterprecis does exactly this. It is therefore only sensible below a chapter.

Epigraph: An epigraph is a smart slogan or motto at the beginning of a chapter. The epigraph environment provides an elegant way of typesetting such a motto. The motto itself (text) and its author (source) are divided by a short line. Unfortunately, we have to fool LyX a bit here again, since the environment needs two arguments (text and source). In this case, we have to use curly brackets (in TeX mode) between the two arguments: *<smart slogan>* `}{` *<author of the slogan>*.

Poemtitle: Memoir has lots of possibilities to typeset poetry (up to very complex figurative poems). LyX can only support a few of them. One is poemtitle, which is a centered title for poems, which will also be added to the table of contents (verse is the standard environment for poems. Memoir has some enhanced versions of verse, but you need to use TeX code, because they have to be nested inside regular verse environments, which is not possible with LyX).

Poemtitle*: Same as poemtitle, but it adds no entry to the table of contents.

[12] Cf. CTAN:/macros/latex/memoir/memman.pdf.

6.16 Article (mwart), book (mwbk) and report (mwrep)

by Tomasz Luczak

The LyX document classes *article (mwart)*, *report (mwrep)* and *book (mwbk)* correspond to the LaTeX document classes `mwart.cls`, `mwrep.cls` and `mwbk.cls`, resp. They are replacements for the standard document classes `article.cls`, `report.cls` and `book.cls`, resp., and fit better to Polish typography conventions in a number of points.

Basic differences:

- Unnumbered titles (with star, e. g. Section*) are added into table of contents,
- Additional page styles:

  uheadings header with separated lines,

  myheadings custom header, contents headers via commands: `\markright` and `\markboth`,

  myuheadings custom header with separated lines,

  outer page number is placed on outer side of page

- Options

  rmheadings serif titles — default,

  sfheadings sansserif titles,

  authortitle on title page first placed is author next title — default,

  titleauthor on title page first placed is title next author,

  withmarginpar reserve place on page for margins.

6.17 Paper

The document class paper provides an alternative to the standard article class. It provides similar functionality, but you might prefer this layout with sans serif sections, headings, and more.

6.18 RevTeX4

by Amir Karger

The Revtex 4 textclass works with the American Physical Sociey's RevTeX 4.0 (the β release of May, 1999) class.

LyX has a Revtex textclass, which works with RevTeX 3.1. However, v3.1 is basically obsolete, as it works with LaTeX 2.09. That means that it doesn't interact very well with LyX, which requires LaTeX 2_ε, although it has been kludged to work. Since RevTeX 4.0 has been designed to work much more cleanly with LaTeX 2_ε, LyX with the RevTeX 4 textclass should also be pretty easy to use.

These documents are supposed to be used in *addition* to the RevTeX 4.0 documents, so we don't describe any of the special RevTeX macros, and assume you'll know what to put in the preamble if necessary.

6.18.1 Installation

All you need to do is install RevTeX 4, as described in the package's README file. The package can be found at The RevTeX 4 Web Site http://publish.aps.org/revtex4/. Install it somewhere that LaTeX can see it. Test it by trying to LaTeX a short RevTeX 4 document in some random directory (i. g. not the directory where you installed the class file.) Then, if you reconfigure LyX, it will find the class file and let you use the RevTeX4 textclass.

Probably the easiest way to get started is either to import a RevTeX 4 document using `tex2lyx`, or to use the Revtex 4 template, found in the templates directory.

6.18.2 Preamble Matter

Optional arguments to `\documentclass`, like "preprint" and "aps", go in the Extra Options field in the Document Layout dialog, as usual. Remember that in RevTeX, at least one optional argument is required!

Other preamble matter, like `\draft` etc. goes in the LaTeX Preamble dialog, also as usual.

6.18.3 Layouts

The layouts basically correspond to the commands in RevTeX4.0. For example, the Email layout corresponds to `\email{}`. Note that (at least as of RevTeX 4.0 Beta), the Address and Affiliation layouts are exactly equivalent, so you shouldn't need to use both.[13]

6.18.4 Important Notes

There are a couple of important unique aspects of RevTeX 4 which might cause bugs that will be even more confusing in LyX.

In RevTeX, the `\thanks` command goes *outside* the `\author` command. The LyX equivalent is that there is a separate Thanks layout. Do *not* write footnotes in the

[13] In case you're curious, both were included so that `tex2lyx` would be able to translate both `\address` and `\affiliation`.

Author layout, or weird things may happen. See the RevTEX 4 documentation for more details.

Also, the Author Email, Author URL, and Thanks layouts must be placed *in between* the Author layout and the corresponding Address (or equivalent Affiliation) layout. If you put the Thanks after the Address, the LATEX won't compile.

6.18.5 Drawbacks

The main problem with this layout is that you can't use the optional arguments to layouts like Email and Title. (The problem is not unique to this layout; you can't use optional arguments to the Section layouts either.) This means that after you export that file to LATEX (which you'll need to do eventually to send it in to APS), you'll need to edit the LATEX file with a text editor to add the optional arguments to set, e. g. the running title for the page headers. Lacking these layouts makes the `\altaffiliation` (and the equivalent `\altaddress`) useless, so the corresponding layouts don't exist, and will have to be added by hand.[14]

6.19 Springer Journals (svjour)

by Martin Vermeer

6.19.1 Description

These are the layout files for some of the journal formats used by Springer Verlag and listed on http://www.springer.de/author/tex/help-journals.html, where you should also go to fetch the class files (yes, these are LATEX 2_ε now!). It is a modular system: the things common to all journals are implemented in `svjour.inc`, which journal-specific layout files (such as, e. g. `svjog.layout` for Journal of Geodesy) can include.

This means that implementing support for any other Springer journal on this list is as simple as writing your own `sv`<myjournal>`.layout` file following the outline given in `svjog.layout`.

It is reasonably well tested only for the Journal of Geodesy. `svjour` and `svjog` come with the standard LyX distribution. Install the relevant class file (downloaded from Springer) in a proper directory, reconfigure LATEX (in the teTEX case by running `texhash`, as root if necessary — doesn't LyX take care of this?), reconfigure LyX and it should work.

6.19.2 New styles

A large number of theorem-like styles — Claim, Conjecture, ... Theorem.

[14] *Note from JMarc:* actually, LyX 1.3.0 supports some forms of optional arguments, but this layout has not been updated yet to take advantage of it.

Headnote, Dedication, Subtitle, Running_LaTeX_Title, Author_Running, Institute, Mail, Offprints, Keywords, Acknowledgements, Acknowledgement. See the Springer class file documentation for details.

6.19.3 Supported journals

- *Journal of Geodesy*: `svjog.layout` — Martin Vermeer
- *Probability Theory and Related Fields*: `svprobth.layout` — Jean-Marc Lasgouttes

Add your own, it isn't so hard!

6.19.4 Credits

These files are partly based on the older `ejour2.layout`, which was again based on a tinkered-with version of an old LaTeX 2.09 style file from Springer. All this, and the `ejour2` layout, are now defunct. Jean-Marc Lasgouttes helped out big in making me find my way around the LyX layout file mechanism.

6.19.5 Bugs

Probably. But probably less than in the old hacked-LaTeX `ejour2`.

Limitations e. g.: does not display the number for theorem-like layouts, just #.

6.20 Slides [aka SliTeX]

by John Weiss

6.20.1 Introduction

This section describes how to use LyX to make slides for overhead projectors. There are two document classes that can do this: the default slides class and the FoilTeX slides class. This section documents the former.

I'm going to say this again, nice and clear, so that there's no misunderstanding:

This section documents the class "slides (default)" *only.*

If you're looking for the documentation for "slides (FoilTeX)", check out section 6.9. The foils class ["slides (FoilTeX)"] is actually somewhat better than the default slides class,[15] which this section documents.

[15] . . . or so I've been told repeatedly by its advocates. Having never used it, I have no idea if this claim is true or not.

This class is the LaTeX 2_ε improvement of the old SliTeX package. Every LaTeX 2_ε distribution includes this class [which I'll just refer to as "slides" from now on], so you're bound to have it. As I noted earlier, there are other classes, such as foils, which also produce slides for overhead projectors and do a better job at it. However, there are some things which slides can do which the others can't, such as generate overlays. Read on to learn more!

6.20.2 Getting Started

Obviously, to use this document class, you need to select "slides (default)" from the class list in the Document ▷ Settings dialog. There are some other special things you should know about this class:

- Don't bother changing the options Sides and Columns. They're not supported by the slides class, anyways.
- The option Page style behaves a bit differently for this class. The possible choices and what they do are as follows:

  plain The final output contains page numbers in the lower right corner.

  headings Like plain, but also prints out any time markers you've put in. This is the default.

  empty The final output contains no page numbers, time markers, or alignment markers.

- The slides class has an extra option: `clock`. To use it, put "`clock`" in the extra class options.

  Using this options allows you to add time markers to Notes. See section 6.20.4.3 for more details.

You can also use the template file "`slides.lyx`" to automatically set up a document to use the slides class [using File ▷ New from Template to open your new document]. The template file also contains some examples of the special paragraph environments used by this class. I'll describe those next.

6.20.3 Paragraph Environments

6.20.3.1 Supported Environments

The first thing you'll notice when you start up a new slides document is the font size and type: it's the equivalent of the size "Largest" in the Sans Serif font. This is also what's used in the output. Think of this as a "visual cue" to remind you that this is a slide. Your final slides will use a larger font; ergo, you'll have less space. Of course, the larger default screen font isn't WYSIWYG, only a reminder.

The next thing that becomes obvious is the changes to the paragraph environment pull-down box [at the far-left end of the toolbar]. Most of the paragraph environments you're used to seeing are missing. There are also five new ones. That's because the slides class itself only supports certain paragraph environments:

- Standard
- Itemize
- Enumerate
- Description
- List
- Quotation
- Quote
- Verse
- Caption
- LyX-Code
- Comment

All of the other standard environments, including the section-heading environments, aren't used in the slides class.

On the other hand, you'll notice the following new environments:

- Slide
- Overlay
- Note
- InvisibleText
- VisibleText

These five are kind of quirky, due to a "feature" in LyX. You see, LyX doesn't permit you to nest any other paragraph environment into an empty environment. Now, that's fine and dandy, but it means that you wouldn't be able to start a slide with anything except plain text. To deal with this, I've performed a little "LaTeX magic."

6.20.3.2 Quirks of the New Environments

All five of the new paragraph environments are somewhat quirky due to inherent limitiations in the current version of LyX. As I just mentioned, LyX forbids environments that begin with another environment. To get around this, the Slide environment isn't a paragraph environment as described in the *User's Guide*.

You should consider Slide, Overlay, and Note to be "pseudo-environments." They look like a section heading or a "Caption," but really begin a [and, if necessary, end the previous] paragraph environment. Likewise, treat InvisibleText and VisibleText as "pseudo-commands." These two perform some action.

A common feature of all five environments, Slide, Overlay, Note, InvisibleText and VisibleText, is a rather long-ish label. The text following this label — ordinarily the contents of the paragraph environment — is utterly irrelevant for Slide, Overlay, Note, InvisibleText and VisibleText. LyX completely ignores it. In fact, you can leave these five environments completely empty.

While you don't *have* to put any text after the rather long-ish label, you might want to. This could be a short description of the contents of the Slide, for example. In that case, enter your descriptive comment and hit Return as you normally would.

If, on the other hand, you don't want to enter any descriptive text, you'll hit another LyX quirk. LyX, like nature, abhors a vacuum, and will not let you start a new paragraph environment until you put something in the old one. So, do this:

- Start entering the text that will *follow* the new Slide, Overlay, Note, InvisibleText or VisibleText.
- Now move to the beginning of that paragraph.
- Next, hit Return.
- Finally, change this new, empty paragraph to a Slide, Overlay, Note, InvisibleText or VisibleText.

Some future version of LyX will, hopefully, resolve this quirkiness...

6.20.4 Making a Presentation with Slide, Overlay and Note

6.20.4.1 Using the Slide Environment

If you're expecting this section to teach you how to actually make a presentation, you'll be sorely disappointed. Naturally, I'll describe all of the ways the slides class can assist you in preparing the materials for a presentation. Filling in the contents, however, is up to you. [Then again, that *is* the LyX philosophy.]

Choosing the Slide environment [in the manner described in section 6.20.3.2] tells LyX to begin a new slide [duh]. The label for this environment/"pseudo-command" is an "ASCII line," in cool blue, followed by the label, "NewSlide:". Any text or paragraph environments that follow this one go on the new slide. It's that simple.

Slides are probably the only time you'll need to forcibly end pages in LyX (this can be specified in the Paragraph Layout dialog). In fact, you'll want to, once you finish entering the contents of one slide. If you've entered more text than can physically fit on a slide, the extra overflows onto a new slide. I don't recommend doing this, however, since the overflow slide won't have any page number on it. Furthermore, it may interfere with any Overlay you've made to accompany the oversized Slide.

The Overlay and Note environments work the same way as the Slide environment. They both create an "ASCII line" followed by a label ["NewOverlay:" and "NewNote:", respectively]. The color is a stunning magenta instead of blue, and the "ASCII line" will look different, in style and in length. The label fonts of all three also differ from one another.

As with a Slide, if the contents of a Note or Overlay exceed the physical size of a slide or sheet of paper, the extra will overflow onto a new sheet. Again, you should avoid this. It defeats the whole purpose of Notes and Overlays.

6.20.4.2 Using Overlay with Slide

The idea behind an Overlay is a slide that sits atop another slide. Perhaps you wish to discuss a figure on the main Slide before displaying the text associated with it. One way to accomplish this is tape a flap of dark paper over the part of the Slide you want to display later. This method fails, however, if you wish to overlap one graph with another, for example. You would then have to fumble while speaking to align the two separate, overlapping Slides to align the two graphs. The use of an Overlay environment in both cases makes life much easier.

Each Overlay receives the page number of its "parent" Slide, appended by "-a".[16] Clearly, you want the contents of both the Slide and the Overlay to each fit on a single physical slide! You should probably consider an Overlay as "part of" a Slide. Indeed, the LyX slides class provides a visual cue for this: the label at the start of an Overlay is shorter than that at the start of a Slide. Lastly, when you generate printable output, you'll find alignment markers in all four corners of both the Overlay page and its parent Slide. These will assist you in lining up the two physical slides.

The major problem in overlaying two slides is aligning the contents of the two transparencies. How much space should you leave for that graph on the second slide? Worse still, what if you want a graph and a sentence on second slide, but there is text on the main transparency that goes in between them? You could try and insert vertical space of the right size. The better way is to use InvisibleText and VisibleText.

As their names imply, InvisibleText and VisibleText are two command-like paragraph environments that make all subsequent text invisible and visible, respectively. Note from section 6.20.3.2 that you don't place anything *into* these two environments, however. When you create an InvisibleText, it inserts a centered, sky-blue label into the page reading "<Invisible Text Follows>". For paragraphs following this label, the

[16] Presumably, mutliple Overlays would have "-a", "-b", "-c", etc. appended to the page number of the parent Slide.

parts of the Slide [or Overlay; it doesn't matter which] where they would be contain instead blank space.

For VisibleText, the corresponding centered label is "<Visible Text Follows>" in blazing green. Paragraphs following this label behave normally. Note that the beginning of a new Slide, Overlay, or Note automatically shuts off an InvisibleText. It's therefore not necessary to use VisibleText at the end of a Slide.

By now, it should be obvious how to create overlay transparencies using the proper combination of InvisibleText and VisibleText on a Slide and Overlay:

1. Create a Slide, including everything that will appear on it, whether on the main slide or on the Overlay.

2. Before each figure or paragraph that will appear only on the Overlay, insert an InvisibleText environment. If necessary, insert a VisibleText environment after the Overlay-only text.

3. Start an Overlay immediately following the Slide.

4. Copy the contents of this Slide into the Overlay.

5. Within the Overlay, change all of the InvisibleText lines to VisibleText and vice-versa.

That's it. You've just made an Overlay.

There's one problem with the way I've designed the LyX slides class: you can't make text in the middle of a paragraph invisible, nor make text in the middle of an invisible paragraph visible again. To accomplish this feat, you'll need to use some inlined LaTeX codes.[17]

6.20.4.3 Using Note with Slide

Like an Overlay, a Note is associated with a "parent" Slide. Here, too, the LyX slides class provides visual cues. The label for a Note is shorter than that of a Slide [yet longer than that of an Overlay] and, like the label of an Overlay is shockingly magenta. Additionally, the printed Note has the page number of its "parent" Slide, appended by "-1", "-2", "-3", etc. You can have multiple Notes associated with a single Slide, and, as with Slide and Overlay, you'll probably want to break up long Notes so that they fit on a single sheet of paper.

[17] The commands of interest are:

- `{\invisible ... }`
- `{\visible ... }`

... and need to be marked as TeX. The text whose "visibility" you wish to change goes in between the brackets [and after the `\invisible` or `\visible` command]. If you don't know how to mark text as TeX, see the appropriate section of the *User's Guide*.

The purpose of a Note is obvious: it contains anything additional you might want to say about a Slide. It could also be used as a sheet of reminders for a particular Slide. In the case of the latter, you might want to make use of time markers. Currently, the LyX slides class has no "native" support for time markers, a SliTeX feature. So, you'll have to resort to using the LaTeX codes.

To use time markers, you'll need to specify the extra class option "`clock`" [see section 6.20.2]. This option turns on timing marks, which will appear in the lower-left-hand corner of every Note you generate. To set what appears in the time marker, you use the LaTeX commands "`\settime{}`" and "`\addtime{}`". The arguments of both commands are time measured in seconds. "`\settime{}`" sets the time marker to a given time. "`\addtime{}`" increments the time marker by the specified amount. Using time markers and Notes in this fashion, you can remind yourself how much time to spend on a particular Slide.

There's one last feature to describe. Clearly, you'd like to print out all of your Slides and Overlays on transparencies while printing all of your Notes on plain paper. However, a Note *must* follow the Slide with which it is associated. What's a person to do?

Luckily, there are two LaTeX commands that allow you to select what to print out. Both must be placed into the preamble of your document. The command "`\onlyslides{\slides}`" will cause the output to contain only the Slides and Overlays. Correspondingly, the command "`\onlynotes{\notes}`" prevents the output of anything but Notes. I'd advise placing both commands in the preamble and initially comment both out. You can then preview your entire presentation as you write. When you're done writing, you can then uncomment one of the two to select what you want to print. I like to uncomment "`\onlyslides{\slides}`" , print to a file with "`-slides`" in its name, comment it back out, then uncomment "`\onlynotes{\notes}`" and print to a "`*-notes.ps`" file. I can then send either file to a printer, loading transparencies or plain paper as appropriate.

You can also provide other arguments to the "`\onlyslides{}`" and "`\onlynotes{}`" commands. See a good LaTeX book for details.

6.20.5 The slides Class Template File

I have also provided a template file, "`slides.lyx`", with the slides class. To use it, begin your new presentation with File ▷ New from Template. Your new LyX presentation file will contain an example Slide – Overlay – Note triplet. The Slide and Overlay additionally contain an example of the use of InvisibleText and VisibleText. Lastly, the preamble will contain:

```
% Uncomment to print out only slides and overlays
%
%\onlyslides{\slides}

% Uncomment to print out only notes
```

```
%
%\onlynotes{\notes}
```

One final thing: I created this class to support the LaTeX 2_ε "SliTeX emulation" class, one of the built-in LaTeX 2_ε classes. Neither I nor the rest of the LyX Team endorse or oppose the use of this built-in slide class. It's here if you want it or need it. There exist other LaTeX 2_ε classes for creating presentations, such as the Foils class [see section 6.9] or the "`seminar`" package [present on some TeX distributions]. The latter is not yet supported under LyX.[18] I know nothing about these other classes. Try them out to see what sort of alternative they provide.

[18] Perhaps you can take on the task...

7 LyX Features needing Extra Software

7.1 Checking TeX

by Asger Alstrup

7.1.1 Introduction

If you have the `chktex` program installed[1], you'll find in the Tools menu the entry: Check TeX. You can get `chktex` it from CTAN, http://www.ctan.org/tex-archive/help/Catalogue/entries/chktex.html.

The ChkTeX package is a program that was written by Jens T. Berger Thiele-mann in frustration because some constructs in LaTeX are sometimes non-intuitive, and easy to forget. The program runs over your LaTeX file, checks the integrity of the file, and flags some common errors. In other technical words, it is `lint` for LaTeX.

Well, what is a syntax checker doing in LyX which is supposed to produce correct LaTeX anyways? The answer is simple: Just as `Lint` not only checks the *syntax* of C programs, but also does *semantic* checks for type-errors, ChkTeX catches some common *typographic* errors, in addition to the syntactical ones. Specifically, ChkTeX is capable of detecting several common errors, such as

- Ellipsis detection:
  Use ... instead of ...
- No space in front of/after parenthesis:
  (wrong spacing)
- Enforcement of normal space after common abbreviations:
  e. g. is too wide spacing.
- Enforcement of end-of-sentence space when the last sentence ends with a capital letter:
  This is a TEST. And this is wrong spacing.
- Space in front of labels and similar commands:
  The label should stick right up to the text to avoid falling to a wrong page. [2]
  The label is separated too much.

[1] `chktex` is not yet available when you are using the LaTeX distribution MiKTeX.
[2] This footnote is in danger of falling off to a wrong page

- Space in front of references, instead of hard spaces:
  In you are in bad luck, the text will break right between the referenced text and reference number, and that's a pity. See section 7.1.1.

- Use of "x" instead of × between numbers:
  2x2 looks cheap compared to 2 × 2.

and more ... It is an invaluable tool when you are "finishing up" your document before printing, and you should run it right after the obligatory spelling check, and before you go fine tuning the typesetting.

7.1.2 How to use it

If you have the program installed, usage is as simple as choosing Tools ▷ Check TEX. This will make LyX generate a LaTeX file of your document, start ChkTeX to check it, and then make LyX insert "error boxes" with the warnings from ChkTeX, if there were any. The warnings will be placed close to the point of the mistake, and you can quickly find them by using the Navigate ▷ Error menu item, or the shortcut key C-g from the default `cua` bind file. Open the error boxes by clicking on them with the mouse, or use the shortcut key C-i from `cua` bindings, or the corresponding C-o for the alternate `emacs` bind file. Read the warning and correct the mistake, if it is a mistake. If you have trouble understanding what the warning is about, you can safely ignore it. Remember that there is a hidden layer between the document on screen and the technical details in invoking ChkTeX, and this gap can make some warnings seem arcane or just right down plain silly.

This document is an excellent testing bed for the feature, and it should provide quite a few warnings for you to fiddle with. Since computers are only so smart, expect most of the warnings to be false alarms, though.

7.1.3 How to fine tune it

Sometimes, you'll find that ChkTeX makes more noise than suits your mood. Then you can choose not to use it, wait until your mood changes, or try to customize ChkTeX to get better along with you.

Although ChkTeX *is* very configurable and extensible, you shouldn't expect to solve all problems with ChkTeX in LyX this way. Since LyX has to generate a somewhat special LaTeX file to be able to match the line numbers from the ChkTeX output[3] to the internal document structure, some of the warnings will not seen to appear correctly. There are two things you can do about this:

- Fine tune the ChkTeX invocation command line in Preferences (tabs Outputs, Misc), or the global ChkTeX installation configuration file (usually with the file

[3] You can inspect the specific output from chktex by using Edit ▷ View LaTeX Log right after a chktex run.

`chktexrc`). See below to learn what warnings can be enabled and disabled on the command line.

- Export your document as a raw LaTeX file using File ▷ Export ▷ LaTeX and run `chktex` manually on that. Invoked in this way, it can be a hassle to find the corresponding place in the document inside LyX, but with a little patience, you should be able to do it.

Here follows the warning messages that can be enabled and disabled in Preferences. Use `-n#` to disable a warning, and `-w#` to enable a warning. The emphasized entries are disabled by default, because the default is "`chktex -n1 -n3 -n6 -n9 -n22 -n25 -n30 -n38`".

Notice that you should only use the options that enable and disable warnings, because LyX relies on some of the other command line parameters to be set in a specific way to have a chance to communicate with `chktex`.

1. *Command terminated with space.*
2. Non-breaking space ("`~`") should have been used.
3. *You should enclose the previous parenthesis with "`{}`".*
4. Italic correction ("`\/`") found in non-italic buffer.
5. Italic correction ("`\/`") found more than once.
6. *No italic correction ("`\/`") found.*
7. Accent command "`cmd`" needs use of "`cmd`".
8. Wrong length of dash may have been used.
9. *"`%s`" expected, found "`%s`".*
10. Solo "`%s`" found.
11. You should use "`%s`" to achieve an ellipsis.
12. Inter-word spacing ("`\ `") should perhaps be used.
13. Inter-sentence spacing ("`\@`") should perhaps be used.
14. Could not find argument for command.
15. No match found for "`%s`".
16. Math mode still on at end of LaTeX file.
17. Number of "`char`" doesn't match the number of "`char`".
18. You should use either `"` or `"` as an alternative to "`"`".

19. You should use "’" (ASCII 39) instead of "´" (ASCII 180).
20. User-specified pattern found.
21. This command might not be intended.
22. *Comment displayed.*
23. Either "\,’ or ’\," will look better.
24. Delete this space to maintain correct page references.
25. *You might wish to put this between a pair of “{}”.*
26. You ought to remove spaces in front of punctuation.
27. Could not execute LaTeX command.
28. Don’t use \/ in front of small punctuation.
29. $\times$ may look prettier here.
30. *Multiple spaces detected in output.*
31. This text may be ignored.
32. Use " to begin quotation, not ’.
33. Use ’ to end quotation, not ".
34. Don’t mix quotes.
35. You should perhaps use “cmd” instead.
36. You should put a space in front of/after parenthesis.
37. You should avoid spaces in front of/after parenthesis.
38. *You should not use punctuation in front of/after quotes.*
39. Double space found.
40. You should put punctuation outside inner/inside display math mode.
41. You ought to not use primitive TeX in LaTeX code.
42. You should remove spaces in front of “%s”
43. “%s” is normally not followed by “%c”.

In later versions of LyX, we hope to provide a more complete interface to this tool (and it’s smaller cousin `lacheck`) to exploit the full power of it. But it’s not exactly useless as it is now: go try it on one of your existing documents of a certain length and be surprised.

7.2 Version Control in LyX

by Lars Gullik Bjønnes and Pavel Sanda

7.2.1 Introduction

A friend of mine wanted to try LyX for a group project. When he didn't find support for version control or file locking, he dropped it. This angered me a bit, so I thought that I should at least make support for RCS (with the possibility of CVS and/or SCCS as a future improvement.) This has been done. LyX now supports some of the most basic RCS/CVS/SVN commands. If you need something a bit more sophisticated you will have to do that manually in a terminal.

Also note that CVS support is not as good as subversion support so we advice using SVN instead. Good place to start with Subversion is SVN Book[4]. In case of RCS you should read "rcsintro" (a man file, read it with `man rcsintro`). This file describes all the basic features of RCS. You should especially notice the comment about a RCS directory, and the notion of a master RCS file (the file ending in `,v`).

Before you begin to use the version control features in LyX, you should be familiar with RCS/CVS/SVN usage. The implementation in LyX assumes a recent version of the GNU RCS or CVS/SVN package—no guarantees are made for older versions. Most of the log messages are not currently displayed after operations — you can check them in Messages pane if unsure. Regular users of version control will appreciate VC toolbar, which can be enabled via View ▷ Toolbars ▷ Version Control..

For introducing your own external commands consult vc-command in the manual of LyX functions.

7.2.2 RCS commands in LyX

The following sections describe the RCS commands supported by LyX. You can find them in the File ▷ Version Control submenu. LyX was tested against RCS 5.7/5.8[5].

7.2.2.1 Register

If your document is not under revision control, this is the only item shown in the menu. And if it is under revision control, the Register item is not visible.

This command registers your document with RCS (unless you are under the directory managed by CVS). You are asked interactively to supply an initial description of the document. The document is now set in Read-Only mode and you have

[4] http://svnbook.red-bean.com/

[5] Note that old DOS / MS Windows builds (5.7) need to setup environment variables in order to work properly with LyX. In particular time zone (TZ) should be set and more importantly RCSINIT in order to change default archive location (`"SET RCSINIT=-x,v/"`) — LyX expects archive to be found in `file.lyx,v` within the original document directory. See also README shipped with Windows RCS binaries.

to Check Out For Edit, before making any changes to it. A document under revision control has a "[RCS:<version> <locker>]" item tagged to the filename in the minibuffer.

RCS command that is run:

```
ci -q -u -i -t-"<initial description>" <file-name>
```

Read `man ci` to understand the switches.

7.2.2.2 Check In Changes

When you are finished editing a file, you check in your changes. When you do this, you are asked for a description of the changes. This is stored in the history log. The version number is bumped, your changes are applied to the master RCS file, the document is unlocked and set to Read-Only mode.

RCS command: `ci -q -u -m"<description>" <file-name>`

7.2.2.3 Check Out For Edit

By doing this you lock the document so that only you can edit it. This will also make the document Read-Write only for you. You will usually continue editing for a while and when you are finished you check in your changes. The status line is changed to reflect that you have locked the file.

RCS command: `co -q -l <file-name>`

7.2.2.4 Revert To Repository Version

This will discard all changes made to the document since the last check in. You get a warning before changes are discarded.

RCS command: `co -f -u<version> <file-name>`

7.2.2.5 Undo Last Checkin

This makes as if the last check in never happened. No changes are made to the document loaded into LyX, but the last version is removed from the master RCS file.

RCS command: `rcs -o<version> <file-name>`

7.2.2.6 Show History

This shows the complete history of the RCS document. The output of `rlog <file-name>` is shown in a browser. See `man rlog` for more info.

7.2.2.7 Revision info

LyX supports RCS version number, author name, date and time of last commit. All those are extracted from `rlog -r <file-name>`. For other details see 7.2.4.9 for details.

7.2.3 CVS commands in LyX

A subset of CVS operations is supported by LyX. You can find the commands in the File ▷ Version Control submenu. The version control system SVN is more powerful, so please use it instead of CVS if possible.

7.2.3.1 General CVS usage

If you start from scratch with CVS you have to create your repository and checkout the working copy with external tools. If you're using a client-server setup you may need to login before doing the first repository checkout.

If your documents are under revision control and others are using the same repository problems arise when different changes to the same document at the same location happen. Standard CVS repositories doesn't operate with a file locking mechanism. This may be surprising, but conflicts only occur if people disagree on the proper content of the same part of a document. So, if co-workers are used to communicate regularly, these conflicts occur rarely. If they don't communicate they have a fundamental problem anyway. Nevertheless some people like to work with so called "reserved checkouts". If they do so the working copy of all files is readonly when checked out first and the user starts editing after using a special command to make the working copy writable. When the changes are checked in the working copy returns to readonly state. With LyX one has to edit the `.cvsrc` file and add the line `cvs -r` to work with reserved checkouts. The benefit is the possibility to see who is using a writable copy of some document. It's not guaranteed only one user makes a copy writable.

LyX tries to guess if you're using reserved or non-reserved checkouts. If your working copy is readonly or it is writable and an additional copy of your document exists in the CVS/Base sub-directory a reserved otherwise a non-reserved checkout is assumed. When a reserved checkout is detected you have to use Check Out to make your working copy writable if it's readonly. After doing so the Check In operation is possible and that makes your working copy readonly again after transferring your changes to the repository.

More information about CVS can be found here `http://www.nongnu.org/cvs` and here `http://ximbiot.com/cvs`.

Read `man cvs` to understand the sub-commands and the switches mentioned below.

7.2.3.2 Register

If your document is not under revision control, this is the only item shown in the menu. And if it is under revision control, the Register item is not visible.

This command registers in CVS your document ***only*** in case you have already the documents directory under CVS control (in particular `CVS/Entries` file exists). This means you have to create or checkout the archive by yourself using external tools. (In case you forget that step LyX registers the document with RCS.)

Then you are asked interactively to supply an initial description of the document. Don't forget that registered file is not yet checked in.

CVS command that is run: `cvs -q add -m"<entered message>" "<file-name>"`

The term `"<file-name>"` above and for all other CVS commands is an abbreviation for "change the current working directory to file location and use the file name without path component as argument".

7.2.3.3 Check In Changes

When you are finished editing a file, you commit your changes. When you do this and you had changed the document, you are asked for a description of the changes. After that changes are written to the repository. In case you didn't change the document and a reserved checkout is detected the reservation made on Check Out is undone.

CVS command:

```
cvs -q commit -m"<description>" "<file-name>" or
cvs -q unedit "<file-name>"
```

7.2.3.4 Check Out Changes

When you are sharing a repository with others, you may have to incorporate their changes into your working copy.

CVS command: `cvs -q update "<file-name>"`

If a readonly checkout is detected the working copy is made writable and reserved.

CVS command: `cvs -q edit "<file-name>"`

7.2.3.5 Revert To Repository Version

This will discard all changes made to the document since the last check in. You get a warning before changes are discarded. Firstly the file is deleted, secondly CVS update command is run.

CVS command: `cvs -q update "<file-name>"`

If a reserved checkout is detected and the working copy has no changes only the reservation is undone.

CVS command: `cvs -q unedit "<file-name>"`

7.2.3.6 Update of the local directory checkout from repository

Once your documents gets more complex, containing sub-documents and pictures, including external `.tex` files and so on using version control becomes more complicated. LyX supports updating the whole tree in which resides the document. This become especially useful once you cooperate with people which neither have detailed knowledge about CVS usage nor they have ambition to commit additional material to the repository. You have to organize the files structure so that all external files are

in the same directory or subdirectories of the document. It's good practice anyway to store multipart documents in an extra directory.

The `Update local directory from repository` command updates the whole directory. If local changes are detected user is warned before update starts. In case of merge conflicts both versions of the conflicting document parts are placed in the final document. You have to review and correct the result of the merge. You'll find the conflicts enclosed in pairs of `<<<<<<<` and `>>>>>>>` separated by `=======`. The first part is your version as before the update operation with the document name prepended. The second one is the repository version with the version number after the sequence of `>` signs.

CVS commands:

`cd $path; cvs diff "."` (Ask if changes are detected.)

`cd $path; cvs -q update "."`

where `$path` stands for the path to the document.

7.2.3.7 Show History

This shows the complete history of the CVS document. The output of `cvs log "<file-name>"` is shown in a browser.

7.2.4 SVN commands in LyX

SVN is now partially supported by LyX. You can find the commands in the File▷ Version Control submenu. Please note that if you use password protected access to repository via ssh, you will be asked in terminal window. LyX was tested against SVN 1.5 and 1.6.[6] Subversion completely changed its internal data structures in version 1.7 and LyX does not support this version yet.

7.2.4.1 Register

If your document is not under revision control, this is the only item shown in the menu. And if it is under revision control, the Register item is not visible.

This command registers in SVN your document ONLY in case you have already the documents directory under SVN control (in particular `.svn/entries` file exists). This means you have to checkout the archive by yourself.

Then you are asked interactively to supply an initial description of the document. Don't forget that registered file is not yet commited.

SVN command that is run: `svn add -q "<file-name>"`

Read `man svn` to understand the switches.

[6] Most of the commands will work with 1.4 too, see 7.2.4.5.

7.2.4.2 Check In Changes

When you are finished editing a file, you commit your changes. When you do this, you are asked for a description of the changes. After that changes are commited.

SVN command:[7] `svn commit -q -m"<description>" <file-name>`

7.2.4.3 Check Out For Edit

Updates the changes of this file from the repository. Be sure you understand SVN merging and conflicts resolving before using this function, because all conflicts has to be resolved manually by you!

SVN command:[8] `svn update --non-interactive "<file-name>"`

7.2.4.4 Revert To Repository Version

This will discard all changes made to the document since the last check in. You get a warning before changes are discarded.

SVN command: `svn revert -q "<file-name>"`

7.2.4.5 Update of the local directory checkout from repository[9]

All the commands above have one shortcomming - they deal with the current document only. Once your document contains pictures, includes external `.tex` files and so on administration becomes more complicated. LyX now supports updating the whole tree in which resides the document[10]. This become especially useful once you cooperate with people which neither know about subversion management nor they have ambition to commit additional material to the repository.

`Update local directory from repository` command updates the whole directory and in case of merge conflicts local version of the files are left, so no unintended data loss occurs. If local changes are detected user is warned before update starts.

SVN commands:

`svn diff $path` (Ask if changes are detected.)
`svn update -accept mine-full $path`

where `$path` stands for the path to the document.

7.2.4.6 Show History

This shows the complete history of the SVN document. The output of `svn log "<file-name>"` is shown in a browser.

[7] In case locking is not enabled. See Section 7.2.4.7.
[8] Ditto.
[9] Note that this command will work only with subversion $\geqq 1.5$
[10] One need to organize the files structure so that all external files are in the same directory or subdirectories of the document.

7.2.4.7 File Locking

The file exchange through various revision control systems brings the problem of merge conflicts in case two different users try to edit the same (parts of) document. When such a conflict happens it needs manual resolving and one reasonable alternative is to provide some kind of locking mechanism, which guarantees that only one user is allowed to edit file at the given time.

SVN has two such mechanisms to provide mutual exclusivity for file access - locks and automatic setting of write permissions (see sec. 7.2.4.8) based on `svn:needs-lock` file svn property[11]. If this property is detected for a given document LyX starts to use SVN locks for document editing automatically and the whole check-in/out mechanism switches to the same regimen as for RCS. This in particular means there are two different modes of file use in LyX:

- Unlocked state. The loaded file is in the read-only mode. For editing on needs to check-out. *Check-out* consists of updating from the repository and gaining write lock. If the lock is not possible to obtain, we remain in unlocked state.

- Locked state. The loaded file is in the 'normal' edit mode. No other user is allowed to edit the file. *Check-in* consists of commiting changes and releasing write-lock. If no changes have been made to the document, no commit will be produced[12] and only the write-lock will be released.

SVN commands:

Check-in: `svn commit -q -m"<description>" "<file-name>"`
`svn unlock "<file-name>"`

Check-out: `svn update "<file-name>"`
`svn lock "<file-name>"`

7.2.4.8 Automatic Locking Property

The above mentioned automatic setting of write permissions of the .lyx file can be set through File ▷ Version Control ▷ Toggle locking property. This command is active only when the file is not locked on the svn server (i.e. you need to check-out before proceeding).

SVN commands:

Set: `svn propset svn:needs-lock ON "<file-name>"`

Unset: `svn propdel svn:needs-lock "<file-name>"`

[11] http://svnbook.red-bean.com/en/1.2/svn.advanced.locking.html
[12] Don't be puzzled by the fact that you will be asked for commit message anyway.

7.2.4.9 Revision Information in Documents

There are more possibilities how to activate revision information in our document.

- LyX supports directly:
  - tree revision information (`vcs-tree-revision`). The result is the output of the `svnversion` command, the following table gives you an idea, how to read the results.

Output	Meaning
4123:4168	mixed revision working copy
4168M	modified working copy
4123S	switched working copy
4123P	partial working copy, from a sparse checkout
4123:4168MS	mixed revision, modified, switched working copy

  - file revision information. The result comes from parsing the output of `svn info -xml file.lyx`. Supported flags are:
    - version number of the last commit (`vcs-revision`)
    - author of the last commit (`vcs-author`)
    - date of the last commit (`vcs-date`)
    - time of the last commit (`vcs-time`)

  You can obtain this info via InsetInfo (e.g. `info-insert buffer vcs-date`). The information will be available only when you have the file stored under svn managment (i.e. the `.svn` directory is available with your document).

- Another—a hacking one—possibility is to use svn keywords[13]. In short – you set file keywords property (e.g. svn propset svn:keywords 'Rev' file.lyx) and then paste keyword TeX code[14] tag in your document (e.g. *Rev*). This way svn client will automatically substitute revision number (e.g. *Rev* : 59) after each update and commit. There are more problems with this approach. Firstly, the '$' character is used in TeX world for math equations, so any occurence of math formula *Rev* become *Rev* : 59 in your LyX document. Similarly for other keywords like Id, Date, Author, etc. Secondly svn output is dependent on your locales, so its very easy that svn would produce some problematic strings once Date is used. Thirdly you get the whole 'Rev: 59' string in your document instead of the plain number. Until subversion implements user's custom keywords it will be hard to use this approach reliably or let LyX to support it directly.

[13] http://svnbook.red-bean.com/en/1.4/svn.advanced.props.special.keywords.html
[14] This is an easy way how to ensure that LyX won't break the line in the middle of keyword tag.

7.2.5 SVN and Windows Environment

> My inclination is to say that if the user cannot figure out the command line operations on their own fairly quickly, they would be well advised to use TortoiseSVN. —P. A. Rubin

7.2.5.1 Preparation

In addition to installing LyX, and having access to a Subversion repository, the user will need to install the Subversion 1.6 client program. A Windows installer for the client program is available from CollabNet. The user may also want to install TortoiseSVN, which integrates Subversion operations into the context (rightclick) menu of Windows Explorer. Operations done outside LyX will typically be more convenient using the Explorer context menu. Note that TortoiseSVN is not a replacement for the client program, which is what LyX itself will use.

7.2.5.2 Bringing a document under Subversion control

Before a LyX document can be brought under version control in Subversion, its parent directory needs to be under version control. If the document is being added to a project already in the repository, this is accomplished by checking the project out to the directory where the new document will be placed. If the project itself is not yet under version control (for instance, if this document starts a new project), the directory must be imported into the repository. This is done outside LyX. Both import and checkout are easily accomplished from the Explorer context menu using TortoiseSVN, or alternatively can be done using the command line client at a DOS prompt. The procedure for importing the project using TortoiseSVN is described below, assuming an existing repository and a new project being started in `C:\new project`. For information on using the Subversion client program, run `svn --help` in a DOS shell.

1. Locate `C:\new project` in Windows Explorer, right click it, and select TortoiseSVN ▷ Repo-browser. If necessary, adjust the URL for the repository, then click OK.

2. Right click the level of the repository under which you want to place the new project folder (typically the top level) and click Create folder Supply a name for the project folder and click OK. Add a message for the log file if desired, then click OK again. The new project folder should appear in the repository. Finally, click OK again to exit the repository browser.

3. Once again right click `C:\new project`, this time selecting SVN Checkout... Select the URL of the project folder you just created in the repository, and set the checkout directory to `C:\new project.` Click OK. You will be warned about a non-empty folder; click OK to proceed. You should now have a `.svn` directory under `C:\new project.`

4. Create or open your document in LyX and click File ▷ Version Control ▷ Register. Add a log message and click OK to commit the document to version control.

From this point onward, you should have full functionality in the File ▷ Version Control menu. You also have the option of checking the document in and out, viewing its history, etc. using the TortoiseSVN context menu in Windows Explorer or the Subversion client program from a command prompt.

7.2.5.3 SSH tunnel used with SVN under Windows

Compared with Linux setting up svn client to communicate over ssh under Windows is a rather troublesome task. We will at least offer some hints how to setup the client side but prior knowledge about ssh and the Windows command line is needed, also be prepared for a great deal of frustration...

1. Get a svn client for windows, as described in the previous sections. When it is a fresh install run some svn command (e.g. `svn -version`) to create config files, which you will need to change later on.

2. Choose a ssh client for Windows. There are several possibilities, we will use the one from Putty tools[15]. You will need to set the connection up so that the client doesn't ask for any password from you. To keep things easy we will use only keys without any additional password protection etc.

   a) Generate keys by `puttygen`. Save your private .ppk key file and put the public one on the server side. If the SVN server runs on Linux, note that the format of the public key is not compatible with Linux openssh and you will need to direcly copy-paste the key from the "*Public key for pasting into OpenSSH authorized_keys file*:" edit field into the server's `~/.ssh/authorized_keys/ authorized_keys2` file.

   b) Get the Putty's `plink`. In the SVN config file[16], section [tunnels], setup ssh command, e.g. `ssh=c:/path/plink.exe -i c:/path/private_key.ppk`.[17]

3. Checkout the SVN archive, e. g.
   `svn co svn+ssh://user@server/repository_path`.

7.2.5.4 End-of-Line Conversions

When the collaborators are mixing Linux and Windows environments, LyX will use different line endings inside the .lyx files. This is not a problem as far as LyX

[15] http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

[16] Exact path depends on Windows version, usually somewhere around `c:\Documents and Settings\user\Application Data\Subversion\config` / `AppData\Roaming\Subversion`.

[17] It will usually take a lot of time to get exact command right and it depends on many things. For example do not have some remote server saved as a default session in Putty etc. If things fail, try to connect via plink without SVN first.

functionality is concerned, but the commit diffs will be huge and merge-conflicts prone. Fortunately SVN itself knows[18] how to deal with CR/LF problems when switching .lyx files to the `native` mode.[19]

7.2.6 Further tuning

With the recent addition of the vc-command function LyX power users are allowed to create their own commands for revision control.

As an example you can see how two TortoiseSVN commands could be integrated directly:

Commit: `vc-command DR "." "TortoiseProc /command:commit /path:$$p"`

Revert: `vc-command DR "." "TortoiseProc /command:revert /path:$$p"`

7.2.7 Version control and Document comparison

One of the typical uses of version control is to inspect the changes between revisions, usually by creating `diff` dumps. While this is useful for plain text files, it is much less useful in the case of LyX files, which have more complicated structure. Hence we provide binding to the Document comparison feature. They are two ways of calling this feature - either by direct call of `VC_COMPARE` LyX function (for details see LyX functions manual) or by icon/menu item Compare with older revision.... One can either compare two chosen revisions of the document or he can simply compare the current version of edited text with older revisions (where '0' revisions back means comparison of the edited file with last commited revision).

This feature is supported for SVN and RCS though due to the more complicated versioning scheme of RCS there is a constraint – when addressing the revisions in dialog, numbers always point to the last number in RCS revision number, i.e. '35' in '1.2.35'.

7.3 Literate Programming

Updated by Kayvan Sylvan (kayvan@sylvan.com), original documentation written by Edmar Wienskoski Jr. (edmar-w-jr@technologist.com)

7.3.1 Introduction

The main purpose of this documentation is to show you how to use LyX for literate programming, where it is assumed that you are familiar with this programming technique, and know what "tangling" and "weaving" means. If that is not the case,

[18] http://svnbook.red-bean.com/en/1.4/svn.advanced.props.file-portability.html
[19] For the lazy guys: `svn propset svn:eol-style native FILE_NAME`

please follow the web links provided in the following sections. There is a lot of good documentation out there covering old development history to the latest tools tips.

It is also assumed that you are familiar with LyX itself to a point that you are comfortable changing your LyX preferences, and X resources file. If that is not the case please refer to other LyX documentation to cover your specific needs.

7.3.2 Literate Programming

From the Literate Programming FAQ:

> Literate programming is the combination of documentation and source together in a fashion suited for reading by human beings. In fact, literate programs should be enjoyable reading, even inviting! (Sorry Bob, I couldn't resist!) In general, literate programs combine source and documentation in a single file. Literate programming tools then parse the file to produce either readable documentation or compilable source. The WEB style of literate programming was created by D. E. Knuth during the development of his TeX typesetting software.

Another excerpt says:

> *How is literate programming different from verbose commenting?*
> There are three distinguishing characteristics. In order of importance, they are:
>
> - flexible order of elaboration
> - automatic support for browsing
> - typeset documentation, especially diagrams and mathematics

Now that I sparked your curiosity, take a look in the references.

7.3.2.1 References

The complete Literate Programming FAQ can be found at:

> Literate Programming FAQ http://www.literateprogramming.com/lpfaq.pdf

The FAQ lists 23 (twenty three!) different literate programming tools. Where some are specialized or "tailored" for particular programming languages, while other have general scope. I selected Noweb for my own use for several reasons:

- It can generate the documentation either in LaTeX or HTML.
- It has a open architecture, i. g. it is easy to plug in new filters and to perform special processing that you may need.
- There is a good selection of filters available already (the HTML is one of them).

- It is free.

The Noweb web page can be found at:

Noweb home page http://www.cs.virginia.edu/~nr/noweb/

Starting from there you can reach many other interesting links and even some literate program examples.

7.3.3 LyX and Literate Programming with Noweb

The LyX support for Literate Programming is provided by using the generic LyX converters mechanism. This support is provided in a "Noweb independent" way, i. g. you will be able to use this new LyX feature with some other literate programming tool of your choice by just changing your LyX preferences.

7.3.3.1 Generating documents and code (weaving and tangling)

Selecting the document class If you have installed Noweb and LyX successfully, whenever you open a new document or try to change the document class of an existing one, you will find that there are three new document classes available:

- Article (Noweb)
- Book (Noweb)
- Report (Noweb)

You must select one of them to create your literate documents from.

Note that literate documents are not limited to these three classes. New classes can be generated from other styles like letter or in combination with other class variations like Article (AMS). If you have special needs that cannot be covered by one of the existing classes, let the LyX developers list (lyx-devel@lists.lyx.org) know and we will arrange to insert a new entry, or teach you how to do it.[20] Moreover, if you use a literate tool other than Noweb you may need to create a new set of document classes for it.

Typing code in LyX enables you to write code with a layout named Scrap.[21] Noweb delimits scraps like this:

[20] It is very simple, it involves the creation of a file with four lines, and re-running of the auto configuration.

[21] The equivalent Noweb term is "Chunk". For historical reasons, I got used to the term "scrap" introduced by other literate tool named Nuweb, which I used for many years before rendering myself to Noweb.

```
<<My scrap>>=
  code
  more code
  even more code
  @
```

The problem is that whatever is written in between the << and the @ must be taken literally, i. g. LyX should be prevented from making any special interpretation of what has been written. This is handled by a special layout named Scrap, that works like a normal paragraph but has a free spacing capability.

The downside of the Scrap paragraph layout is that consecutive paragraphs of code will be spaced with one empty line in the source code and also in the printed documentation. The work around is to enter each line of code within a single Scrap, with a newline (ctrl-return). The example above will look like this:[22]

```
<<My scrap>>=
  code
  more code
  even more code
  @
```

This layout works fine. The only real inconvenience is that you have to type ctrl-return instead of a plain return.[23]

As a special note, you can also use the "%def" construct of Noweb in your scraps to add items to Noweb's identifier cross-reference:

```
<<My scrap>>=
  def some_function(args):
    "This is the doc string for this function."
    print "My args: ", args
@ %def some_function
```

For an example of this usage and the resulting cross-reference output, look at the Literate python program in *LIBDIR/examples/listerrors.lyx* which should make this all clear.

Generating the documentation At this point you already have a new document file with a proper document class, and with some code and text on it. How do I print it? The answer is simple, you select View ▷ DVI, etc. Just like you would do for a plain document. No special procedure is required.

To help orientate you, I will now explain what happens inside LyX:

[22] If you have a printed version of this document you will not see any difference between the previous example and this one.

[23] It is in my list of "improvements" to fix that.

1. When the Update ▷ DVI menu option is chosen, a LaTeX file is generated.

   If the document is of any literate class the generated file will be named with an extension name defined by the "literate" format (defined in the Preferences panel), otherwise the file will have the usual `.tex` extension.

2. Note that the only difference so far is in the name of the file, no special processing is required by LyX. Given that you formatted the code using the Scrap layout that, by itself, takes care of the business.

3. If the document is of any literate class LyX will then use the internal LyX to Noweb converter, followed by the Noweb to LaTeX converter[24] to generate the LaTeX file.

   Otherwise it will just skip this step.

4. Finally, LaTeX is invoked and the regular post processing continues as in a plain document.

Independence from a particular "literate tool" is easily achieved by changing the commands that are run by the various converters.

Generating the code When the build menu option is chosen or the corresponding button in the toolbar is pressed, a LaTeX file is generated just like step 1 above. Next, LyX invokes the `Noweb->Program` converter. This converter needs to be defined by the user and is not installed by default, though the Program format is. This converter (like any other converter) will have two parts:

1. The converter program itself. This program performs the conversion from the one format to the other (in this case, from the Noweb format to the Program pseudo-format).

2. The error log parser. This is a program whose sole purpose is to rewrite error messages in a format that LyX understands. This makes it possible for LyX to place error boxes in the right places in the file buffer.

The first part, the "Converter" setting, should be set to "`build-script $$i` $$r". This basically means that LyX will call "build-script" (a program or script) with the name of the Noweb file (normally a file in the LyX temp directory) and the directory path of the original LyX file.

This is an implementation of "build-script" that you can place in a directory on your path:

```
#!/bin/sh
#
```

[24] The converters are defined in the Tools ▷ Preferences panel, under the "Conversion" tab. See section *Converters* of the *Customization* manual for general information about converters.

notangle -Rbuild-script $1 | env NOWEB_SOURCE=$1 NOWEB_OUTPUT_DIR=$2 sh

The next part of the converter setting is the "Flags" which is to be set to "`parselog=$$s/scripts/listerrors`". This will run any errors that are generated by the "build-script" process through the "listerrors" program.

The build will normally take place in LyX's temporary directory, so the files produced by the conversion will be in that directory. LyX will copy out what it regards as the 'main' file, but the `Noweb->Program` conversion may produce several files, and so most of these would then be deleted when LyX was closed. This is why we pass in the NOWEB_OUTPUT_DIR environment variable so that the build-script scrap can place the generated files in that location.

Build instructions in the document The last piece of the integration between LyX and noweb is the "build-script" scrap. Generally, the instructions for building your program should be embedded in a scrap of its own. The noweb-specific "build-script" above uses the notangle command to look for this scrap (called "build-script") and runs its contents through "sh".

Typically, such a scrap would look something like this:

```
<<build-script>>=
#!/bin/sh

if [ -z "${NOWEB_SOURCE}" ]
then
  NOWEB_SOURCE=myfile.nw
fi
[... code to extract files ... use NOWEB_OUTPUT_DIR here ...]
[... code to compile files ...]
@
```

Look in *LIBDIR/examples/listerrors.lyx* or in *LIBDIR/examples/Literate.lyx* which implement two versions of the "listerrors" program for some illustrations of how all of these pieces go together or in *LIBDIR/examples/noweb2lyx.lyx.* Interestingly, these three files show off the language-indepence of the LyX literate programming support since they are written in Python, C and Perl respectively.

7.3.3.2 Configuring LyX

All the Literate Programming support is configured by the Tools ▷ Preferences panel in the "File Handling" tab. The important parts are:

the "NoWeb" format Set up via the File Formats tab, this is where the Noweb-specific pieces are set up. The GUI Name is set to `NoWeb`, the file extension is set to `.nw`. This tells LyX to create a file with a `.nw` extension in the first step of the conversion process.

the Program format This is an empty format whose sole purpose is to be the endpoint of a conversion (which then allows us to set up a converter for it).

NoWeb->LaTeX This converter performs the "weaving" of the literate document. For Noweb, it is set to "`noweave -delay -index $$i > $$o`"

NoWeb->Program This performs the "tangling step". As stated above, the Converter is set to "`build-script $$i` $$r", with Flags set to "`parselog=$$s/scripts/listerrors`".

7.3.3.3 Debug extensions

There is also a new function implemented in the LyX server, the "server-goto-file-row" function, to be used with ddd/gdb or other debugger.

When debugging code with ddd/gdb, it is possible to invoke a text editor at the current execution position with a single key stroke. The default ddd configuration for that is shift-ctrl-V. It happens that you can define the editor command line invocation in ddd by accessing the Edit ▷ Preferences ▷ Helpers dialog and changing the "Edit Sources" entry.

I take advantage of the newly created LyX server function and this ddd feature, and set "Edit Sources" to:

```
echo "LYXCMD:monitor:server-goto-file-row:@FILE@ @LINE@" >~/.lyxpipe.in
```

With this, whenever you are using ddd and find a point in the program that you want to edit, you just press shift-ctrl-V (in the ddd window), and ddd will forward this information to LyX through the LyX server and then the LyX window will show the same file with the cursor at the same position ddd was pointing to. No more guessing or long scrolling to locate a point in the program back from debugging !

Note however that you must enable the LyX server to get this feature working (it is disabled by default). You can enable it in Preferences (tabs Inputs, Paths) by entering in the LyXserver pipe a path like "`/home/<your-home-directory>/.lyx/lyxpipe`"

Read the LyX server documentation in the *Customization Manual* for further information.

7.3.3.4 Toolbar extensions

There are six new buttons that can be added to your LyX toolbar. Five of these buttons are short cuts to layout styles: Standard, Section, LaTeX, LyX-Code, and Scrap. The last one is a short cut to the "Build Program" File menu entry.

LyX has a range of buttons that are available for tool bar customization. In my toolbar I like to combine the six short cuts above with two more: One for View ▷ Update ▷ DVI and the other for View ▷ DVI File menu entries. Here is how it looks like:

```
Toolbar
  Layouts
```

```
  Icon "layout Standard"
  Icon "layout Section"
  Icon "layout LaTeX"
  Icon "layout LyX-Code"
  Icon "layout Scrap"
  Separator
  Icon "buffer-view"
  Icon "buffer-typeset"
  Icon "build-program"
  Separator
.
.
.
End
```

7.3.3.5 Colors customization

There are a number of colors in LyX that can be customized in Preferences. One of the things that bothers people is the LaTeX font color. The default color is red, since the scraps uses LaTeX font, and there is a lot of scraps in literate documents, you may get tired of seeing everything in red. You can change it by going to the tabs Look&Feel, Colors.

The next thing is the visible presence of the newline character in the screen. You can choose the color of this particular character and make it blend in the background. I recommend you choosing a color that is close to the background but not equal, that way you still can see it is there, but it is not bothering you anymore.

7.3.4 LyX and knitr/Sweave

Support for knitr and Sweave is documented in the knitr.lyx example file and in the Sweave manual (in the Help > Specific Manuals submenu).

8 Special features

You can do everything with LyX that you can do with LaTeX. However, LyX cannot support every LaTeX feature directly. But you can always use TeX Code. This chapter shows you some more special things you might want to use. For other special things you can do with figures, tables, floats, boxes and notes, have a look at the *EmbeddedObjects* manual.

8.1 Multiple Columns

The aim for this section is to show how the LaTeX package **multicol** can be used in a LyX document. LyX does not support the **multicol** package natively yet, so you have to use some TeX code.

The **multicol** package allows switching between one and multicolumn format on the same page. Footnotes are handled correctly, but will be placed at the bottom of the page and not under each column. Within the different columns you can use everyting, with the limitation that for floats you need to use the option Span colums for each column.

8.1.1 Two columns

If you want to have two columns in your text, insert the comand
\begin{multicols}{2}
at the point where you want the two column layout to start, and then
\end{multicols}
where you want it to end. Like this:

The Adventure of the Empty House
by Sir Arthur Conan Doyle

It was in the spring of the year 1894 that all London was interested, and the fashionable world dismayed, by the murder of the Honourable Ronald Adair under most unusual and inexplicable circumstances. The public has already learned those particulars of the crime which came out in the police investigation, but a good deal was suppressed upon that occasion, since the case for the prosecution was so overwhelmingly strong that it was not necessary to bring forward all the facts. Only now, at the end of nearly ten years, am I allowed to supply those missing links which make up the whole of that remarkable chain. The crime was of interest in itself, but that interest was as nothing to me compared to the inconceivable sequel, which afforded me the greatest shock and surprise of any event in my adventurous life. Even now, after this long interval, I find myself thrilling as I think of it, and feeling once more that sudden flood of

joy, amazement, and incredulity which utterly submerged my mind. Let me say to that public, which has shown some interest in those glimpses which I have occasionally given them of the thoughts and actions of a very remarkable man, that they are not to blame me if I have not shared my knowledge with them, for I should have considered it my first duty to do so, had I not been barred by a positive prohibition from his own lips, which was only withdrawn upon the third of last month.

8.1.2 Multiple columns

The same pattern is used when you want more than two columns:

It can be imagined that my close intimacy with Sherlock Holmes had interested me deeply in crime, and that after his disappearance I never failed to read with care the various problems which came before the public. And I even attempted, more than once, for my own private satisfaction, to employ his methods in their solution, though with indifferent success. There was none, however, which appealed to me like this tragedy of Ronald Adair. As I read the evidence at the inquest, which led up to a verdict of willful murder against some person or persons unknown, I realized more clearly than I had ever done the loss which the community had sustained by the death of Sherlock Holmes. There were points about this strange business which would, I was sure, have specially appealed to him, and the efforts of the police would have been supplemented, or more probably anticipated, by the trained observation and the alert mind of the first criminal agent in Europe. All day, as I drove upon my round, I turned over the case in my mind and found no explanation which appeared to me to be adequate. At the risk of telling a twice-told tale, I will recapitulate the facts as they were known to the public at the conclusion of the inquest.

You can have more than 3 columns if you want to, but that might not be very pleasant for the eye.

8.1.3 Columns inside columns

You can even have columns inside columns:

The Honourable Ronald Adair was the second son of the Earl of Maynooth, at that time governor of one of the Australian colonies. Adair's mother had returned from Australia to undergo the operation for cataract, and she, her son Ronald, and her daughter Hilda were living together at 427 Park Lane.

The youth moved in the best society–had, so far as was known, no enemies and no particular vices. He had been engaged to Miss Edith Woodley, of Carstairs, but the engagement had been broken off by mutual consent some months before, and there was no sign that it had left any very profound feeling behind it. For the rest {sic} the man's life moved in a narrow and conventional circle, for his habits were quiet and his nature unemotional. Yet it was upon this easy-going young aristocrat that death came, in most strange and unexpected form, between the hours of ten and eleven-twenty on the night of March 30, 1894.

Ronald Adair was fond of cards–playing continually, but never for such stakes as would hurt him. He was a member of the Baldwin, the Cavendish, and the Bagatelle card clubs. It was shown that, after dinner on the day of his death, he had played a rubber of whist at the latter club. He had also played there in the afternoon. The evidence of those who had played with him– Mr. Murray, Sir John Hardy, and Colonel Moran–showed that the game was whist, and that there was a fairly equal fall of the cards. Adair might have lost five pounds, but not more. His fortune was a considerable one, and such a loss could not in any way affect him. He had played nearly every day at one club or other, but he was a cautious player, and usually rose a winner. It came out in evidence that, in partnership with Colonel Moran, he had actually won as much as four hundred and twenty pounds in a sitting, some weeks before, from Godfrey Milner and Lord Balmoral. So much for his recent history as it came out at the inquest.

Please read the example file `multicol.lyx` for more advanced examples including column and header spacing, vertical separator lines, and more.

8.2 Non-standard Paragraph Shapes

There are times
when the tyranny
of rectangular
paragraphs must
be overthrown. In
such situations, a
call to the delightful
plain TEX command
`\parshape` is called for.
As you can see, completely
arbitrary shapes can be laid
out with a suitable set of
linelength definitions.
While this parshape
may look a bit silly
and useless, one
could conceive of
situations such as
finely tuned dropped
capitals, word
wrapping around
non-rectangular
graphics, etc. which
will benefit from
such handcrafting.

The syntax is `\parshape numlines #1indent #1length #2indent #2length ... #nindent #nlength`, where `numlines` is the number of lines of text which define the paragraph. If there turn out to be fewer lines, the shape is truncated; if there are more, the excess lines have the same dimensions as the last line of the definition. The `#nindent` and `#nlength` entries specify the indentation of the line from the left margin, and the length of the line as measured from that point. The shape applies only to the current paragraph; everything is reset to normal for the next paragraph.